Maria Francesca Costabile
Yvonne Dittrich
Gerhard Fischer
Antonio Piccinno (Eds.)

LNCS 6654

End-User Development

Third International Symposium, IS-EUD 2011
Torre Canne (BR), Italy, June 2011
Proceedings

Springer

Lecture Notes in Computer Science 6654

Maria Francesca Costabile
Yvonne Dittrich
Gerhard Fischer
Antonio Piccinno (Eds.)

End-User Development

Third International Symposium, IS-EUD 2011
Torre Canne (BR), Italy, June 7-10, 2011
Proceedings

Springer

Volume Editors

Maria Francesca Costabile
Università di Bari Aldo Moro
70125 Bari, Italy
E-mail: costabile@di.uniba.it

Yvonne Dittrich
IT University of Copenhagen
2300 Copenhagen, Denmark
E-mail: ydi@itu.dk

Gerhard Fischer
University of Colorado at Boulder
Boulder, CO 80309-0430, USA
E-mail: gerhard@colorado.edu

Antonio Piccinno
Università di Bari Aldo Moro
70125 Bari, Italy
E-mail: piccinno@di.uniba.it

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-21529-2 e-ISBN 978-3-642-21530-8
DOI 10.1007/978-3-642-21530-8
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011928546

CR Subject Classification (1998): D.2, D.1, I.7, K.6, K.8

LNCS Sublibrary: SL 2 – Programming and Software Engineering

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The number of computer users keeps growing as a result of the wide spread of information and communication technology in everyday work and life. Computer systems functionality and presentation need to cater to a growing variety of use situations and interests. With that, end users are evolving from being passive software consumers to acquiring a more active role as developers and producers. This evolution is triggered by several factors, including: (1) the deployment of innovative technologies and designs like Web 2.0 technologies and service-oriented architectures that support people to not only use software, but also create it; and (2) the increasing importance of a global infrastructure, particularly the mutual dependencies between computer-based tools, work practices, domain competencies and organizations. These developments require a differentiation of roles beyond the conventional user-designer dichotomy.

End-user development (EUD) refers to methods, techniques, and tools that support end users to create, adapt or evolve software artifacts. Many applications already support some EUD activities, ranging from simple parameter customization to modification and assembly of components, creating simulations, games and Web content. To provide engaged professionals in all domains with tools to develop their own applications has been a vision from the early days of software engineering that motivated the development of high-level, visual, and domain-oriented programming environments. To make this vision a reality has been the core objective of EUD.

Practices of EUD, however, differ depending on purpose, context, and technologies. Different requirements and challenges have to be addressed when providing support, e.g.: (1) the development of mashups supporting leisure activities; (2) systems supporting the admission process of a university that need to take legal requirements into account; (3) tools that are to be used as a common resource by different users; (4) or mobile applications that run on small handheld devices. EUD brings together research on technical innovations, human–computer interaction, organizational aspects, and the investigation of cooperation among end-user developers with professional designers. The selection of articles in this volume indicates that the challenge is no longer to prove that EUD tools and techniques are possible, but to understand how to support EUD by taking different contexts into account.

The Third International Symposium on EUD brought together researchers and practitioners from industry and academia working in the field of EUD. Participants met for four days in Torre Canne (Brindisi), a lovely small resort on the beautiful Adriatic coast in southern Italy. They came from more than 15 countries in the world, including some very far away, like Brazil and New Zealand. The rich and exciting technical program consisted of presentations of accepted papers, two keynote speeches, a panel, the Doctoral Consortium and three workshops.

Fourteen long papers and 21 short papers, which were carefully selected by the International Program Committee, were in the program; they range from meta-design approaches, methodology and guidelines, to designing frameworks for end-user applications, enabling EUD through mashups, providing infrastructures, up to discussing legal aspects of EUD. Their presentation at the symposium was organized into sessions whose titles reflect the chapter organization in these proceedings.

The two keynote speakers, both renowned researchers, greatly contributed to the high-quality program. John Bacus, Product Manager at Google Inc., USA, gave the opening keynote. Fabio Casati, Professor at the University of Trento, Italy, was the presenter of the closing keynote.

The program of the main symposium also featured a panel titled EUD: From Opportunity to Challenge. The panel was organized and moderated by Boris De Ruyter, Principal Scientist at Philips Research Europe, The Netherlands, and the panelists explored EUD developments and their impact by taking needs and opportunities from industry and from academia into account.

The Doctoral Consortium was organized by Daniela Fogli of the University of Brescia, Italy, and Elisa Giaccardi of Carlos III University of Madrid, Spain. It was held on June 7, the day before the main symposium. Fourteen papers of PhD students were accepted and are included in these proceedings. An award in memory of Piero Mussio (University of Milan, Italy), who was among the first researchers working in the field of EUD, was awarded to the PhD student presenting the most interesting and innovative research.

Anne-Marie Kanstrup of Aalborg University, Denmark, and Anders Morch of University of Oslo, Norway, were the Workshop Co-chairs. Brief descriptions of the three challenging workshops, held in parallel with the Doctoral Consortium on June 7, are included in the final part of these proceedings.

We are very grateful to all those who contributed to the success of IS-EUD 2001, including the authors, the International Program Committee, and the Steering Committee. Special thanks go to the other members of the Organizing Committee: Paolo Buono and Rosa Lanzilotti of the University of Bari, Italy, who did a great job as Publicity Co-chairs and also designed and managed the website; Carmelo Ardito of the University of Bari, Italy, who served as Local Chair. Finally, we thank the University of Bari for the resources provided to support the organization of the Third International Symposium on EUD.

June 2011

Maria Francesca Costabile
Gerhard Fischer
Yvonne Dittrich
Antonio Piccinno

Organization

General Chairs

Maria Francesca Costabile	University of Bari, Italy
Gerhard Fischer	University of Colorado, USA

Program Chairs

Yvonne Dittrich	University of Copenhagen, Denmark
Antonio Piccinno	University of Bari, Italy

Workshop Chairs

Anne-Marie Kanstrup	University of Aalborg, Denmark
Anders Mørch	University of Oslo, Norway

Doctoral Consortium Chairs

Daniela Fogli	University of Brescia, Italy
Elisa Giaccardi	Carlos III University of Madrid, Spain

Publicity Chairs

Paolo Buono	University of Bari, Italy
Rosa Lanzilotti	University of Bari, Italy

Local Chair

Carmelo Ardito	University of Bari, Italy

Steering Committee

Boris de Ruyter	Philips Research, The Netherlands
Volkmar Pipek	University of Siegen, Germany
Mary Beth Rosson	Pennsylvania State University, USA
Volker Wulf	University of Siegen, Germany

Program Committee

Michael Atwood	Drexel University, USA
John Bacus	Google Inc., USA
Jörg Beringer	SAP Research, USA
Paolo Bottoni	Sapienza University of Rome, Italy
Margaret Burnett	Oregon State University, USA
Danilo Caivano	University of Bari, Italy
John M. Carroll	The Pennsylvania State University, USA
Ellen Christiansen	Aalborg University, Denmark
Vincenzo D'Andrea	University of Trento, Italy
Clarisse De Souza	PUC-Rio, Brazil
Cleidson De Souza	IBM Research, Brazil
Paloma Diaz	Carlos III University of Madrid, Spain
Jeanette Eriksson	BTH, Sweden
Athula Ginige	University of Western Sydney, Australia
Thomas Andreas Herrmann	University of Dortmund, Germany
Heinrich Hussmann	University of Munich, Germany
Kari Kuutti	University of Oulu, Finland
Catherine Letondal	ENAC/LII, France
Henry Lieberman	MIT, USA
Agostino Marengo	University of Bari, Italy
Gary Marsden	University of Cape Town, South Africa
Nikolay Mehandjiev	University of Manchester, UK
Sebastian Ortiz-Chamorro	National University of La Plata, Argentina
Sharon Oviatt	Incaa Designs, USA
Philippe Palanque	ICS-IRIT, Paul Sabatier University, France
Cecile Paris	CSIRO ICT Centre, Australia
Nandish V. Patel	Brunel University, UK
Fabio Paternò	CNR-ISTI, Italy
Samuli Pekkola	Tampere University of Technology, Finland
David Redmiles	University of California, USA
Alexander Repenning	University of Colorado, USA
Mitchel Resnick	MIT, USA
Stefan Sauer	University of Paderborn, Germany
Judith Segal	The Open University, UK
Helen Sharp	The Open University, UK
Carla Simone	University of Milano-Bicocca, Italy
John Thomas	IBM T. J. Watson Research Center, USA
Genoveffa Tortora	University of Salerno, Italy
Michael Twidale	University of Illinois, USA
Corrado Aaron Visaggio	University of Sannio, Italy
Jacob Winther	Microsoft Dynamics, Denmark
Yunwen Ye	Software Research Associates Inc., Japan

Additional Reviewers

Balagtas-Fernandez, Florence
Bortolaso, Christophe
Cao, Jill
Chong, Ming Ki
Daughtry, John
Du, Honglu
Hoffman, Blaine
Jiang, Hao
Koehne, Benjamin
Kulesza, Todd
Latzina, Markus
Maurer, Max-Emanuel
Nolte, Alexander
Piorkowski, David
Prilla, Michael
Roy Chowdhury, Soudip
Shinsel, Amber
Turnwald, Marc
Valtolina, Stefano
Winckler, Marco

Table of Contents

Part I: Keynote Speeches

Part II: Long Papers

Mashups

Frameworks

Users as Co-Designers

Infrastructures

Methodologies and Guidelines

Beyond the Desktop

Part III: Short Papers

Mashups

Frameworks

Users as Co-Designers

Methodologies and Guidelines

Beyond the Desktop

End-User Development in the Workplace

Meta-Design

Supporting End-User Developers

Part IV: Doctoral Consortium

Part V: Workshops

Part I
Keynote Speeches

End-User Development at Scale: Real-World Experience with Product Development for a Large and Engaged User Community

John Bacus

Google Inc., 2590 Pearl Street #100,
Boulder, CO 80302, USA
jbacus@google.com

Abstract. Google SketchUp is a 3D modeling tool that anyone can learn to use, and they do so in unprecedented numbers for an application of its type. Its populist position in the often esoteric world of 3D modeling has in no small part been enabled by the shared wants, needs and contributions of a large, vocal and committed user base. Unlike many more insularly built software applications, SketchUp's development team cultivated this community of active user contributors in tandem with the software engineering effort right from the very earliest releases. Initially, this community grew in user forums and was concerned primarily with user-to-user training and troubleshooting. However, in recent years the SketchUp team has been experimenting with deeper end-user development in other areas of our product.

In this talk, I will share experiences and some success metrics in three specific project areas. First, Google's "Your World in 3D" initiative, which is attempting to crowd-source the construction of a 3D model of every building in the world. Second, I will describe the design and implementation of SketchUp's Ruby API and its use by the user community to extend SketchUp with powerful but specialized plugins. And finally, I will share experiences and success (and failure!) metrics around our team's continued involvement in community cultivation through forums for the design, specification and prioritization of new feature development in the SketchUp client.

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, p. 3, 2011.

How End-User Development Will Save Composition Technologies from Their Continuing Failures

Fabio Casati

Department of Information Engineering and Computer Science
University of Trento
firstname.lastname@unitn.it

Abstract. In this paper I discuss the motivation behind the continuing failure of composition technologies (workflows, service composition, mashups)over the past two decades, as well as the repeated mistakes we (the IT community, including companies) keep making. I then argue that, despite this, we do need these technologies, now more than ever, and that EUD is the driver behind this need, so the non-programmer has to be the target user for these platforms. Next, I share ideas as well as research results on how to design and develop composition tools that can be "programmed" by end user to define fairly complex logic both in the case of process-oriented applications and of mashups. These ideas are instantiated in a set of principles as well as in composition languages and tools developed in cooperation with our research and industrial partners.

1 Failure of Composition Technologies

Models and systems that compose work items, services and applications have been around for decades. In the late eighties and early nineties, workflow systems became popular as a mechanism to automate office work and its flow from employee to employee. Later on, as services (and web services) started to appear, workflow languages and systems evolved from orchestrating *people* to orchestrating *services*. More recently, with the advent of (Web) components rich in UI and API available for reuse, mashups surfaced as the latest buzzword and technology[1].

The majority of the platforms supporting composition technologies have the common characteristic of providing a graphical language for specifying the composition logic. Another common characteristic is that they claim to be oriented to *end users*. This is said more or less explicitly in the thousands of papers and manuals describing such systems. Ease of use and understandability by end-users and business people are often quoted as the main reason for adopting these technologies. Indeed, there is hardly a reason to provide charming graphical environments when programmers have shown over and over that they like to type when they code, rather than draw shapes with a mouse, which is really impractical whenever the logic becomes anything more than trivial to define. Furthermore, the complexity of the specifications for many such

[1] I will use the term composition tool/technology to generically refer to workflows, service composition, and mashup tools/technologies.

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 4–6, 2011.

composition languages makes the claim of end-user orientation risible. Yet, complexity is there because real life compositions are complex.

Another very common aspect of all these graphical composition models and tools is their *failure*. Every time there is a new component technology, gzillions of papers, conferences, and IT platforms begin to appear, developed by large and small companies. And then they disappear. We have seen this over and over with workflows and service composition, and now the same has happened with mashups, where Popfly was only the first example. Indeed graphical process notation are used for essentially two purposes today: for (informal,non-executable) *modeling* of processes, typically as a tool for process understanding or requirements specification, and for *monitoring*, where an abstracted version of the actual process is animated to show managers or end-users the status of the execution. But only rarely are they used for specifying any sort of executable behavior.

Yet, we should not be discouraged by these problems and failures[2]. Today we do need to empower end users to develop applications. We see this in many aspects of our daily life. There are all sorts of consumer services, widgets and devices available that as users we do feel we want to use and combine, perhaps in a *situational* fashion. And at the workplace we see how frequently processes change. For these applications there is no point in going through a traditional software engineering lifecycle: it is too slow and too expensive in a landscape where processes, services and technologies change quickly and are short-lived. Therefore, we do need composition and we do need end-user composition in particular. So the questionis not *if* we need EUD for composition, but how to achieve it, learning from decades of failures (with no stopping in sight so far), and building on top of many recent studies on the topic that tells us what novices can and cannot do [3,6,7,8].

2 End-User Composition

The talk that this paper summarizes, and that is available in[2], discusses how we can achieve EUD for composition. Briefly, the underlying principles and ideas are the following.First, composition tools and languages today aim at being *generic* (domain-independent), *powerful*, and *simple*. This is a very bold proposition which is likely to fail, and indeed it did fail. We need to give up something. And this something is generality. We cannot live without ease of use and we cannot live with languages that lets us only model toy behaviors, because real life is not like that.But if we go domain specific, then we can design languages that speak the language of the user. Indeed domain-specific process management tools, such as ETL tools, are among the few to have succeeded.

Second, we need to get rid of complex process logic. There is no place for such concepts as the BPMN-style gateways in EUD. This cannot however be in contrast with the principle of allowing complex logic, which is needed.

Third, because complex problems require complex solutions, the complexity needs to be somewhere. We argue that the complexity needs to reside in the components, never in the composition logic, so that composition remains simple. This might seem obvious but is not the road that has been followed thus far.

[2] I made a similar claim in a book I wrote [1] talking about hopes for service composition, and I was wrong. But I am an optimist.

Fourth, we need to get rid of data mappings (and maybe of data as well). We cannot expect end users to write XPath expressions or anyways transform data from a format to another, probably not even if done "by example", and it is far fetched to expect users to read the "human-readable" XML.We need to find a way for components to be able to communicate – and so to be somewhat "homogeneous" - without the need for data transformation.

Finally, we need to assist the developers step-by-step during the process both by leveraging development knowledge from the crowd (of developers) [4,5] and by providing simulation of the outcomes while they are developing. This is both to make them *feel* what comes out of the composition and because we want the community to help make developers aware of the opportunities in a given domain, which would be difficult to discover otherwise. EUD enables this because there is potentially a large number of people reusing the same components in a given domain.

In the talk [2], besides detailing these principles, we go through a set of proposals that can realize them and as a consequence make EUD feasible for composition-based applications. In a nutshell, the idea is based on making the tools domain-specific, which fixes the overall data concepts and models, in imposing that all components can always process (at least with a *pass-through* semantics) all data elements so that data mappings are never needed, and in providing composition paradigms that have the same complexity of the widely used and accepted informal flowcharts (to specify process-oriented behaviors) or of the juxtaposition of components (to specify UI synchronization-based integration behaviors). We also see how these ideas are put at work in concrete domains, models, and tools.

Acknowledgments. Florian Daniel, Maurizio Marchese, Heng Chang, Soudeep-Chowdhuri, Antonella De Angeli, Muhammad Imran, Carlos Rodrigues, Stefano Soi contributed a lot to the ideas presented here and in the referenced talk. This work was partially supported by the European Commission (FP7 projects OMELETTE, no. 257635; andLIQUIDPUB, no. 213360).

References

1. Alonso, et al.: Web Services. Springer, Heidelberg (2004)
2. Casati, F.: How End-User Development Will Save Composition Technologies from their Constant Failures, Available from `http://mashart.org`
3. Recker, J., Safrudin, N., Rosemann, M.: How novices model business processes. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 29–44. Springer, Heidelberg (2010)
4. Chowdhury, R., et al.: Wisdom-Aware Computing: On the Interactive Recommendation of Composition Knowledge. In: WESOA 2010 (2010)
5. Greenshpan, O., Milo, T., Polyzotis, N.: Autocompletion for Mashups. In: VLDB 2009 (2009)
6. Namoun, A., Nestler, T., DeAngeli, A.: Service Composition for Non Programmers: Prospects, Problems, and Design Recommendations. In: ECOWS 2010 (2010)
7. Wong, J., Hong, J.I.: Making mashups with marmite: towards end-user programming for the web. In: CHI 2007 (2007)
8. Mehandjiev, N., Lecue, F., Wajid, U., Namoun, A.: Assisted Service Composition for End-Users. In: ECOWS 2010 (2010)

Part II
Long Papers

Enabling End User Development through Mashups: Requirements, Abstractions and Innovation Toolkits

Cinzia Cappiello[1], Florian Daniel[2], Maristella Matera[1],
Matteo Picozzi[1], and Michael Weiss[3]

[1] Politecnico di Milano, Dipartimento di Elettronica e Informazione
P. zza L. da Vinci, 32 - 20134 Milano, Italy
{Cinzia.Cappiello,Maristella.Matera,Matteo.Picozzi}@polimi.it
[2] University of Trento
Via Sommarive 14, 38123 Trento, Italy
daniel@disi.unitn.it
[3] Carleton University
1125 Colonel By Dr, Ottawa, ON K1S 5B6, Canada
weiss@sce.carleton.ca

Abstract. The development of modern Web 2.0 applications is increasingly characterized by the involvement of end users with typically limited programming skills. In particular, an emerging practice is the development of *web mashups*, i.e., applications based on the composition of contents and functions that are accessible via the Web. In this article, we try to explain the ingredients that are needed for end users to become mashup developers, namely adequate *mashup tools* and *lightweight development processes*, leveraging on the users' capability to *innovate*. We also describe our own solution, the *DashMash* platform, an example of end-user-oriented mashup platform that tries to fill the gaps that typically prevent end users from fully exploiting the mashup potential as innovation instruments. DashMash offers an intelligible, easy-to-use composition paradigm that enables even inexperienced users to compose own mashups. As confirmed by a user-centric experiment, its paradigm is effective and increases the satisfaction of the end users.

Keywords: Web Mashups, End User Development, User-driven Innovation.

1 Introduction

The current trend in the development of modern web applications – and in particular of those applications commonly referred to as Web 2.0 applications – clearly points toward a high user involvement. One of the emerging "user-intensive" practices today is the development of online applications starting from contents and functionality that are available on the Web in form of open APIs or reusable services. A "classical" example is www.housingmaps.com, which interweaves housing offers taken from the Craigslist with Google Maps. The phenomenon is commonly known as ***web mashups***, and it shows that web users are increasingly also taking part in the *development*

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 9–24, 2011.

process of web applications, in addition to taking part in the content *creation process* like in social web applications (e.g., Wikipedia).

The use of ***open services*** is a unique feature that distinguishes mashup development from other (component-oriented or service-based) development paradigms. Currently, the most popular mashups integrate public programmable APIs (like Google Maps and the Twitter API), but also RSS/Atom feeds (e.g., stock news), content wrapped from third party web sites (e.g., product prices), or any kind of available Web services providing computing support or acting as plain data sources [25]. However, the vision is that of so-called *enterprise mashups* [15], a porting of current mashup approaches to company intranets, enabling enterprise members to play with a company's internal services that give access to the enterprise information assets, and to mash them up in innovative, hopefully value-generating ways, for example, to automate a recurrent bureaucratic procedure.

Provided that suitable tools and methodologies for mashup composition are available, through these open services (both public and company-internal services) even less skilled end users could evolve from passive receivers of innovation to actors actively involved in the ***creation of innovation***. Aggregated over all users, this speeds up innovation (as users conduct parallel experiments with the same service), and covers a wider range of the design space than the service providers could have achieved on their own, had they not exposed their services to other parties. The effort that almost all of the big players of today's Internet economy (e.g., IBM, Intel, Yahoo!, SAP, etc.) are investing into research on mashups is indeed a clear indicator that there is something going on, which goes beyond the current "hacking" of mashups on the Web.

Despite this great potential, there is however a lack of adequate tools and methodologies that really empower the end user to compose services and innovate. In this article, we explore the mashup world, its potential as a tool to be offered to end users to create innovation and its current limits, and propose a new solution through which end users can easily create mashups. In Section 1, we shortly introduce the mashup world and explain why end users are interested in doing their own applications and who else benefits from this practice. Guided by our experience in the development of mashup tools, and by some experimental results, we then discuss the mashup development process and derive a set of requirements that mashups should meet, in order for end users to be able to use them profitably (Section 2). Next, we describe our tool, *DashMash*, that has been conceived to enable users to easily compose mashups supporting analytical processes and, hence, to innovate (Section 3), and in Section 0 we report on a user evaluation of DashMash. In Section 4 we discuss related work, and in Section 5 we finally draw our conclusions.

2 Rationale and Background

Web mashups support the "composition" of applications starting from services and contents oftentimes provided by third parties and made available on the Web. Mashups were initially conceived in the context of the consumer Web, as a means for users to create their own applications starting from public programmable APIs, such as

Google Maps or the Twitter API, or contents taken from Web pages[1]. However, the vision is towards the development of more critical applications, for example the so-called enterprise mashups [15], through which enterprise users can compose by themselves and in a flexible way their dashboards for process and data analysis, using the plethora of available corporate services (e.g., for the access to a variety of enterprise information sources), Web resources and open services. Mashups are therefore gaining momentum as a technology for the creation of innovative solutions in any context where flexibility and task variability become a dominant requirement. A "culture of participation" [10], in which users evolve from passive consumers of applications to active co-creators of new ideas, knowledge, and products, is indeed more and more gaining momentum [24].

2.1 User-Based Innovation and Innovation Toolkits

There is a specific driver at the heart of the mashup phenomenon and user participation: *user innovation*, i.e., the desire and capability of users to develop their own things, to realize their own ideas, and to express their own creativity. In a traditional design-build-evaluate cycle, feedback from the user is only collected once a product prototype has been developed. Thus feedback is collected late, and changes to the product that reflect an improved understanding of customer requirements are costly. In a user-driven innovation approach, a service provider offers users an ***innovation toolkit*** through which users can build their own products [24]. This toolkit provides a constrained interface on the capabilities of the company's product platform, but this ensures that the new products are properly constructed, adhering to a sort of *conservative invention* [12].

In general, the idea behind an innovation toolkit is that the iterative experimentation needed to develop a new product can now be entirely carried out by the user. Many users can work in parallel on the solution to a problem, by focusing on their own version of the problem. They can create a solution that closely meets their needs and can more quickly obtain feedback from their development experiments. At the same time, the toolkit provider does not carry the cost of failed experiments. Nonetheless, if an experiment turns out to add significant value, the company can integrate the user innovation back into its core product. On the Web, this is what happened when developers mashed up Flickr with maps. Subsequently, Flickr has incorporated a map function into both its platform and public service. Google also monitors the use of its public APIs (such as Google Maps and Google Search) to fine-tune the APIs and to learn from the best innovative uses [14].

2.2 End Users Involvement in the Mashup Development Scenario

The way in which mashups are developed depends on the type of mashup. While current *consumer mashups* (for example, all the numerous mashups based on Google Maps) are mainly the results of some hacking activities by expert developers, *enterprise mashups* highlight different potential scenarios that might involve users at

[1] The Web site www.porgrammableweb.com manages a repository of consumer mashups.

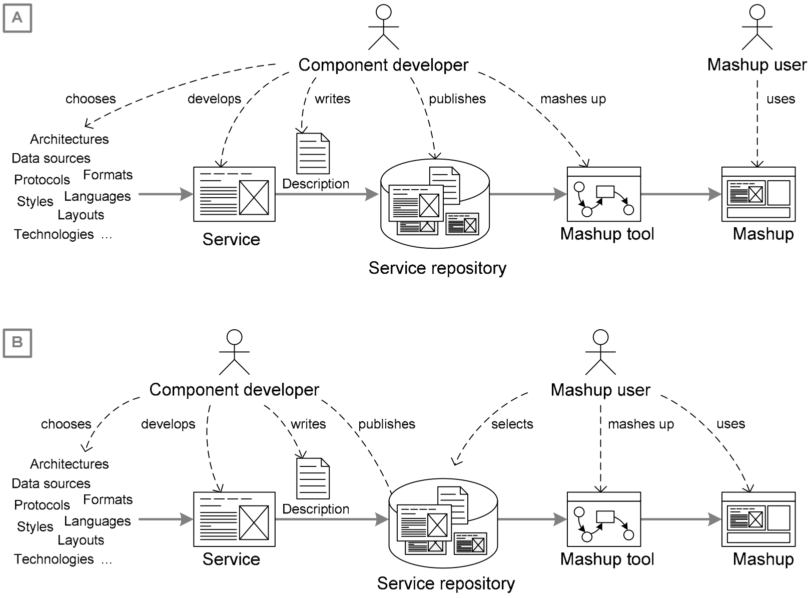


Fig. 1. The mashup development scenarios

different skill levels [19]. In the enterprise context it is indeed possible to recognize two main situations:

A) Mashup tools can be used by ***expert developers*** (for example implementers of an IT department or service providers) to deliver applications quickly. End users are not directly involved in the construction of such mashups but benefit from the shorter turn-around time for new applications. The resources for developing mashups are limited to the expert developers in the IT department. Given the limited resources of an IT department, only frequently requested applications will be developed.

B) Expert developers create services in a format that can be more easily consumed and combined into mashups by ***users who are not themselves developers***, for example requiring simple parameterizations of components; they also provide a tool where anyone creates their own mashups. This is analogous to how spreadsheets are used in organizations today: end users (e.g., business analysts) can create spreadsheets without involvement from an IT department. These mashups are often created for a single purpose and user (they are indeed also known as situational applications [1]), thus they potentially address a larger diversity of user needs.

Fig. 1 illustrates the previous scenarios. The two (extreme) corresponding solutions differ in terms of the heterogeneity of the services that can be combined, the diversity of user needs that can be met, and the level of sophistication of either the user or the tools that support their work. A tool for the creation of mashups (scenario B) will, initially, be the most challenging scenario to implement. However, it also provides the biggest pay-off. Using the tool, users can combine services and data to create their own mashups. The tool constrains what users can do and, hence, ensures the composability of mashup components. In the sense of the earlier discussion on user innovation [24], such a tool provides a toolkit that enables users to create their own applications. However, users are not limited in terms of the types of applications they can build: this scenario, therefore, supports the greatest diversity of user needs.

Another distinction between the two scenarios is the degree of control over the ***quality*** of mashups being created. In scenario A, the IT department fully controls what kind of mashup is being developed. Thus, the IT department ensures the quality of those mashups. However, not all mashups have stringent requirements in terms of security, performance, or reliability; they may only be used for a specific purpose, and a complex solution developed by the IT department would also be too costly. In scenario B, the IT department selects which components can be mashed up and provides an environment for safely executing those mashups. Users can create mashups from those components to meet needs unanticipated or not served by the IT department. Such mashups may subsequently serve as prototypes for hardened applications developed by the IT department, should there be a need for the mashup to be exposed to many users within the enterprise, or if the mashup has to be offered to outside users.

3 The Need for Lightweight Development Processes

Based on the previous observations, it derives that the ideal mashup development process should reflect the innovation potential of mashups: to *compose* an application, starting from given *contents and functionality* responding to personal needs, and to simply *run* it, without worrying about what happens behind the scenes. The *prototype-centric* and *iterative* approach that in the last years has characterized the development of modern Web applications is even more accentuated: the composer, i.e., the mashup end user, just mashes up some services and runs the result to check whether it works and responds to his needs. In case of unsatisfactory results, he fixes the problems and is immediately able to run the mashup again. The following requirements, which also characterize the EUD domain [7,10], emerge as fundamental ingredients enabling the end user composition of mashups:

– ***Domain-specific focus*****:** In order to allow users to make sense of the services and components that are available for composition, it is important to customize the platform with respect to well-defined domains the user is comfortable with. In a mashup environment, this can be achieved through mechanisms for the easy integration into the platform of public or internal services, adhering to possibly different technologies and standards, that handle domain-specific requirements meeting the user needs.

- ***Abstraction from technical details:*** In order to help users understand the features provided by the available services and the effect that each service may have on the overall composition, we need to develop a tool that *speaks the language of the user*, both in terms of functionalities and terminology known to the user. We therefore need metaphors to represent services and service coupling that abstract from technical details, e.g., their programmatic interface or communication protocol. Users should be asked to manipulate, e.g., add, remove, or modify, *visual objects* by operating service visualization properties rather than being required to configure technical details of services and the composition logic. As also confirmed by our user-centric experiment (see Section 0) this increases user satisfaction and, in particular, the perceived control over the composition process.
- ***Continuous feedback:*** In order to further enhance the users' perception of the effects that individual actions or services have on the final applications and to allow users to understand the current state and look&feel of the composition, it is highly desirable to provide *immediate visual feedback* on any composition action and to support the *immediate execution of the resulting mashup*. This requirement is backed by our observations that show that end users typically have difficulties in understanding the difference between design time and runtime.
- ***Composition support:*** In order to achieve a tool that speaks the language of the user, it is also important to aid those users that *don't speak the language of the tool*, that is, those users that do not have sufficient development knowledge. Composition can be *assisted or guided* in multiple ways, for instance, by providing recommendations of compatible services that can also increase the quality of the final mashup [21], of composition patterns that have been used successfully in the past [22], or also by pre-compiling or automatically connecting services on behalf of the user (see the next section).

While there are many mashup tools or platforms available today, none of these addresses all the above requirements, which we however regard as fundamental ingredients if we really want to enable end users to develop their own applications.

4 The DashMash Platform for Sentiment Analysis

The development of a mashup environment responding to the needs highlighted in the previous section is the object of our own research on the agile, lightweight development of mashups. The environment is called *DashMash*, it is an evolution of our prior work on mashup composition [27], and aims at an integration approach where a variety of different component types and technologies, ranging from simple RSS feeds to complex SOAP or RESTful Web services and UI components[2] can be integrated into the platform and then combined by the end-users, thanks to the adoption of some descriptive models for both component services and mashup composition.

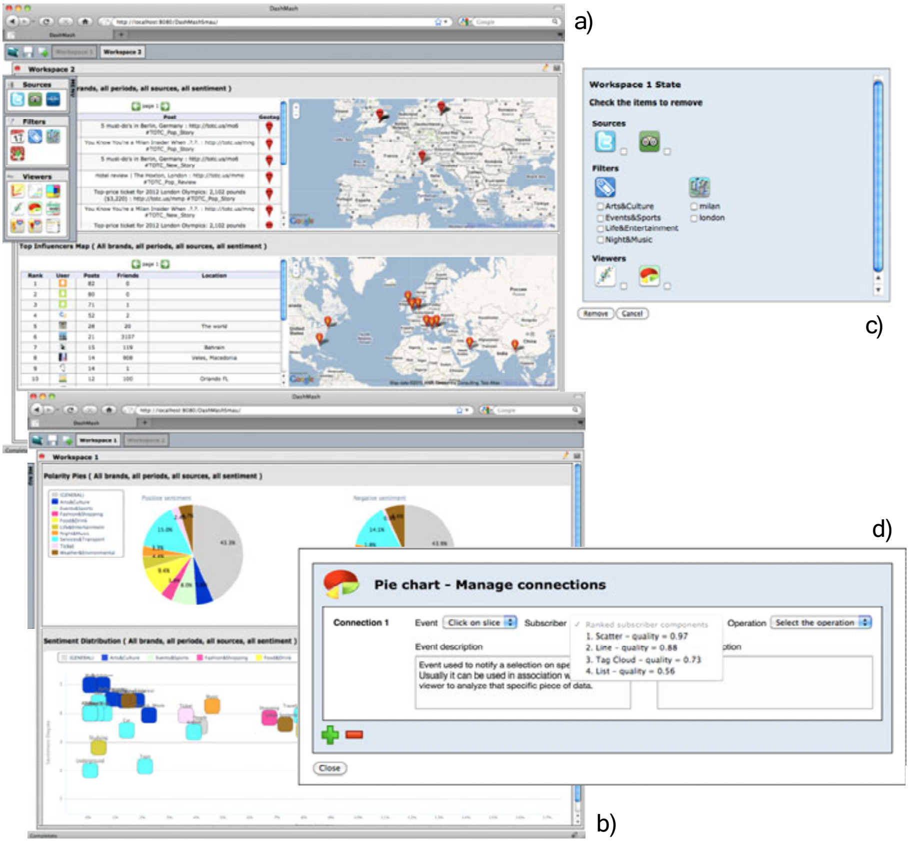
[2] UI components are characterized by a presentation level (the User Interface) that is reused "as is" within the final integrated mashup. Google Maps is an example of UI component: beside its application logics related to geo-localization, it also offers a UI for the map-based visualization of geo-localized data.

DashMash is a mashup tool, specifically conceived for the construction of dashboards exploiting both company-internal services extracting data from local data warehouses, and public APIs and web resources. Recently DashMash has been specialized for sentiment analysis (the ***domain***), an emerging business intelligence practice that aims at understanding market trends from the unsolicited feedback provided by user comments published on the Web through social applications. An ongoing project funded by the Municipality of Milan focuses on the design of an engine that is able to automatically extract sentiment indicators summarizing the opinions contained in user generated contents [2]. In this context, DashMash has been adopted to allow end users, i.e., analysts and decision makers interested in improving the quality of services offered by Milan city, to "compose" their analysis flexibly, playing in variable ways with sentiment indicators, and also complementing such indicators with interesting external Web resources, for example linking sentiment indicators to news, events, and opinions that cause trends and behaviors. The DashMash customization to sentiment analysis has required the development of some ad-hoc services for the sentiment indicators computation and visualization, which are offered to the users as basic, still configurable, elements for their compositions.

As shown in Fig. 2, mashup creation is enabled through a web-based, visual environment; the visual composition paradigm has been specifically conceived to hide the complexity of the technical details and the composition languages actually managing the execution of the mashup (the ***abstraction***). As shown in Fig. 2(a), a visual menu at the left hand side presents the list of services: *data sources* that materialize contents extracted from community sites, several types of *filters*, a multiplicity of *viewers* to visualize data, which are both open APIs, e.g., the Google APIs for maps and charts, ad-hoc developed services[3], and utility open APIs/services, such as RSS feeds and calendars. Each component is denoted through an icon and a label that shortly recall the offered functionality. Components can be mashed up by moving their corresponding icons into the so-called *workspaces*. As soon as a component is moved into a workspace, its UI is immediately rendered so that the users can easily check whether the component choice satisfies their needs.

Each workspace is associated with a data set, which results from the integration of data sources and filters that the users can select and configure depending on their needs. Some default rules also assume that in absence of user selections some data sources are automatically associated with the Workspace. In this way, the creation of meaningful mashups is preserved. Each workspace visualizes its data set according to the visualizations offered by selected *viewers*. For example, Fig. 2(a) shows a mashup in which the user has selected two data sources, storing contents extracted from two social applications, Twitter and TripAdvisor, and has filtered them by using a keyword-based filter, with key = "Milan". Contents are then presented through a pie chart viewer, visualizing the percentage of comments related to categories of interest in the tourism domain (e.g., food, entertainment, art, and other relevant entities), and a scatter plot visualizing the average value of sentiment for the same set of categories.

[3] Several viewers offering graphic visualizations have been developed using the Highcharts JS library (http://www.highcharts.com/), to offer advanced presentations specific for sentiment indicators.


a)
DashMash
Workspace 1
Workspace 2
Sources
Filters
Viewers
Top Influencers Map (All brands, all periods, all sources, all sentiment)
Rank
User
Posts
Friends
Location
Polarity Pies (All brands, all periods, all sources, all sentiment)
Positive sentiment
Negative sentiment
Sentiment Distribution (All brands, all periods, all sources, all sentiment)
c)
d)
b)

Workspace 1 State

Check the items to remove

Sources

□ □

Filters

□ Arts&Culture
□ Events&Sports
□ Life&Entertainment
□ Night&Music

□ milan
□ london

Viewers

□ □

Pie chart - Manage connections

Connection 1 Event: Click on slice Subscriber

✓ Ranked subscriber components
1. Scatter - quality = 0.97
2. Line - quality = 0.88
3. Tag Cloud - quality = 0.73
4. List - quality = 0.56

Operation: Select the operation

Event description

Event used to notify a selection on spe
Usually it can be used in association w
viewer to analyze that specific piece of data.

Fig. 2. The DashMash editor for mashup composition and immediate execution. The two screenshots show the mashup of sentiment analysis dashboards [2, 6].

Fig. 2(b) shows a second mashup defined on top of the same data sources as the previous one. In this case, the filters select comments from users that are considered opinion leaders, the so-called *influencers*, who are visualized through a *list viewer* integrated with Google Maps to show the influencers' location. This is an example of integration between an internal service (providing information about influencers) and a public API, this latter providing an added value to the overall analysis.

Users can iteratively modify the composition, by adding or dropping components through some visual actions. Changes are enacted in real time, i.e., the mashup visualization changes accordingly, so that users can immediately see the effect of their composition actions in their workspace (the ***continuous feedback***). They can also access a detailed description of the status of the current composition (see Fig. 2(c)), summarizing the main elements, their configuration and synchronization behavior, and easily modify sources, filters, viewers or even configuration properties of single filters or viewers.

Once a new component is added, the system automatically binds the component to the ones already in the mashup (the ***composition support***). The platform can automatically generate service bindings, based on a service classification and on corresponding parameter-operation couplings. For example, when a new viewer is added into a workspace, its visualization logic is automatically mashed up with the corresponding data sources and filters associated with the workspace. Users can then introduce further synchronization behaviors. Simple dialog boxes, abstracting from technical details, allow them to create new service combinations resulting in synchronized behaviors. For example, starting from the mashup shown in Fig. 2(a), the dialog box presented in Fig. 2(d) allows the user to set a coupling so that a click on a pie slice contextualizes the analysis offered by the map viewer to that selected label. Based on descriptive models of components, the dialog box presents possible connection points, namely the component events (see next section), exposed by the components selected by the user, plus a short description of the resulting synchronization behavior. The system provides suggestions about other candidate components based on compatibility rules and quality criteria [21].

The rest of this section is devoted to illustrate the architectural elements and the mechanisms that implement the previous functions and behaviors in DashMash.

4.1 DashMash Architecture

The overall organization of the DashMash platform is illustrated in Fig. 3. The mashup execution is centered on a lightweight paradigm in which the orchestration of registered services, the so-called *components*, is handled by an intermediary framework in charge of managing both the definition of the mashup composition and the execution of the composition itself. Different from the majority of mashup platforms, where mashup design is separate from mashup execution, in DashMash the two phases strictly interweave. The result is that composition actions are automatically translated into models describing the composition, and these models are immediately executed. Users are therefore able to interactively and iteratively define and try their composition, without being forced to manage complicated languages or even ad-hoc visual notations.

Mashup Execution. DashMash capitalizes on the mashup paradigm defined in [27], which is based on an event-driven model operating at the presentation level: *events* generated from the user interaction with one mashup component (e.g., the selection of a slice in a pie chart) can be mapped to *operations* of one or more components subscribed to such events (e.g., the visualization of details of the selected data in a scatter plot). The occurrence of events, intercepted by an *Event Broker* module, causes a state change in the subscribed components. Each component therefore keeps running according to its own application logic, within the scope defined by an HTML <*div*>. As soon as events occur, the involved components publish them. Based on the definition of service binding, the so-called *listeners,* an *Execution Handler* then notifies the subscribed components, if any, and triggers the execution of their corresponding operations.

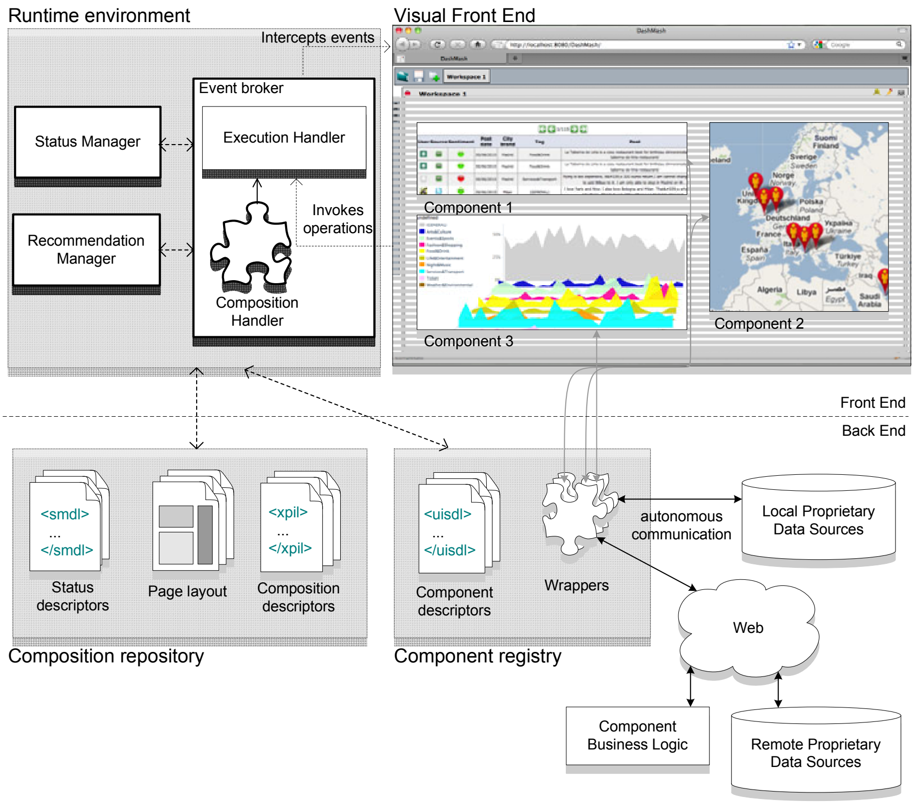


Fig. 3. Overall organization of the DashMash platform

Listeners are specified in a *composition model*, expressed by means of the XPIL (eXtensible Presentation Integration Language) XML-based language [27]. This composition logic also requires each component to be characterized by a model expressing the binding with the actual service/API, the events that the component can generate, and the operations that enable other components to modify its internal state. This component description, expressed by means of the UISDL (UI Service Description Language) XML-based language [27], provides a uniform model to coordinate the mashup composition and execution, which obviates the heterogeneity of service standards and formats by embedding only the information needed for synchronizing services at the presentation level. The adoption of such a component model is an important ingredient toward the provision of an environment where technical details are hidden to the user.

Component and composition models are stored in dedicated repositories:

- The *Composition Repository* maintains the XPIL-based specifications of the compositions as created by the users, the HTML templates for the mashup layout management, and a *state model* that maintains information about the configuration of a mashup instance (i.e., values instantiating parameters and specific configuration of the involved components), to support saving and restoring functions, history management, and also the easy, “on-the-fly” modification of the composition (as shown in Fig. 2(c)).

- The *Component Registry* stores the component descriptive models plus wrappers through which the platform invokes service operations. The creation of component wrappers is the only “technical” activity that is required to register services into the DashMash platform, and, as such, it is up to the expert developer, not the end user. However, once the component registration is performed, the end user can transparently use and integrate any service through the visual paradigm illustrated above.

Mashup composition and automatic model generation. Due to the intermixing between mashup composition and execution, in DashMash events captured by the Event Broker can be related not only to users and system actions occurring during the mashup execution (those ones managed by the Execution Handler, which causes a change to some component’s state), but also to the dynamic definition of the composition (e.g., the drag&drop of a component icon into the composition area). The Event Broker intercepts events and dispatches them to modules in charge of their handling. In particular, the *Composition Handler* manages composition events. It automatically translates the addition of a component into a set of listeners, based on default couplings between the involved services. Based on such listeners, it creates or updates (if already existing) the current composition model. It also dispatches the composition events to the *Status Manager* in charge of maintaining the description of the mashup instance status. As soon as the composition and the status update are complete, the mashup composition is reloaded and immediately rendered through the visual front-end. The mashup is then executed according to the event-driven, publish-subscribe logic that characterizes the Execution Handler.

Service binding definition. DashMash supports the definition of *default* and *custom* bindings:

- *Default bindings* are automatically defined by the Composition Handler when a composition action is intercepted and ensure a minimum level of inter-component synchronization that does not require end users to explicitly define service coupling. To enable the automatic definition of default bindings, we start from a classification of components. For example, in order to facilitate the construction of a dashboard, it is possible to identify four component classes, namely *data services*, retrieving data from corporate/relational data sources, *filters*, expressing selection conditions over the context defined by a workspace, *viewers*, supporting the visualization of result sets also offering data aggregation and transformation functions, and *generic components*, i.e., any kind of open service (local or remote) offering functionality that can make the analysis process more effective. Service classification is domain-specific, and needs to be revised for any DashMash customization. Classification changes, however, only imply a new configuration of the Composition Handler, while no other changes are required to other architectural elements.
- *Custom bindings* are user-defined. Nevertheless, the Composition Handler supports the user in the choice of components and component bindings, since it generates compatibility– and quality– based recommendations. To this aim, it dispatches the composition events to the *Recommendation Manager*, an

additional module of the runtime environment that is in charge of evaluating the quality of the current composition and providing suggestions about the selection of possible components to add to or of compatible components that can substitute the existing ones in order to achieve or improve the mashup quality [5, 21].

5 Validation

In order to validate the composition paradigm of DashMash with respect to user needs, we conducted a study involving 35 participants. Six of them were real end users of the DashMash sentiment analysis customization, i.e., analysts and decision makers external from our research institution, that are supposed to actually use DashMash for their analyses, with a medium technical expertise. Other users were master students of the Computer Engineering program at Politecnico di Milano, featuring different levels of technical background: 12 of them were already acquainted with concepts related to service composition and mashups. The others were familiar with Web application development but not with service composition and mashups. Prior to the experiment, none of them was exposed to the tool and to our research results.

We observed users completing two tasks through DashMash, which consisted in the composition of mashups extracting and visualizing data related to two specific sentiment indicators: the first mashup aimed at showing the percentage of volume for the positive and negative sentiment along different brand categories; the second mashup was related to the volume distribution in time for the positive sentiment. In both the mashups, multiple components needed to be coupled, also by means of custom bindings for which users were required to define new listeners. Our goal was to assess how easily the users would be able to develop a composite application. The experiment specifically focused on the effectiveness and intuitiveness of the composition paradigm, trying to measure such factors in terms of user performance, ease of use and satisfaction.

We expected all users to be able to complete some experimental tasks, with however a greater efficiency (e.g., reduced completion task times) and a more positive attitude (in terms of perceived usefulness, acceptability and confidence with the tool) by expert users. Their domain knowledge and background could indeed facilitate the comprehension of the experimental tasks, and improve the perception of the control over the composition method, and thus, their general satisfaction. However, surprisingly no significant differences in task completion time were found between experts and novices. In particular, domain expertise was not discriminating for task 1 ($p = .085$) and for task 2 ($p = .165$). The average time for task 1 completion was indeed 154 seconds for domain experts and 215 seconds for domain non-experts. Similarly, technology expertise was not discriminating for task 1 ($p = .161$) and for task 2 ($p = .156$) - in average 200 seconds for experts vs. 232 seconds for non-experts. The lack of significant differences between the two groups does not necessarily mean that expert users performed badly. However, it indicates that the tool enables even inexperienced users to complete a task in a limited time and that the expertise needed to properly understand the necessary concepts and to operate the tool is relatively low.

Another interesting result is that the difference in completion times for the two tasks is about half a minute ($t = 28.2$, $p = .017$), i.e., a reduction of about 15%. This result highlights the learnability of the tool [13]: although the second task was more critical compared to the first one, subjects were able to accomplish it in a shorter time.

The ease of use was confirmed by the data collected through four questions in the post-questionnaire, asking users to judge whether they found it easy to identify and include services in the composition, to define service bindings between services, and to monitor and modify the status of the mashups. On average, users gave the ease of use a mark of 1.77 (the scale was from 1 - very positive to 7 - very negative). The distribution ranged from 1 to 4 (*mean = 1.77, meanS.E. = .12*). We did not found differences between novice and expert users. This was especially true for the perceived usefulness ($p = .51$).

The post-experiment questionnaire also allowed us to assess the user satisfaction by means of a semantic differential scale requiring users to judge the method on 12 items. We did not find significant differences between experts and novices. Despite our initial assumption, we therefore found that the ease of use of the tool is perceived in the same way by novice and expert users, although the latter have greater domain knowledge. Moreover, the moderate correlation between the satisfaction index and the ease of use index ($\rho = .55$, $p = .011$) also reveals that who perceived the method as easy also tended to evaluate it as more satisfying. This confirms that ease of use is perceived.

6 Related Works

So far the research on mashups has focused on enabling technologies and standards, with little attention on easing the mashup development process - in many cases mashup creation still involves the manual programming of the service integration. There is a considerable body of research on mashup tools, the so-called mashup makers, which provide graphical user interfaces for combining mashup services, without requiring users to write code. Among the most prominent platforms, Yahoo!Pipes (http://pipes.yahoo.com) focuses on data integration via RSS or Atom feeds, and offers a data-flow composition language. JackBe Presto (http://www. jackbe.com/) also adopts a pipes-like approach for data mashups, and allows a portal-like aggregation of UI widgets (mashlets). MashArt [8] focuses on the integration of heterogeneous components (not only data or RSS feeds), offering a mashup design paradigm through which composers create graph-based models representing the mashup composition.

With respect to manual programming, all the previous platforms certainly alleviate the mashup composition tasks. However, to some extent they still require the user to deeply understand the application logic behind services and the integration logic. In some cases, building a complete Web application also equipped with a user interface requires the adoption of additional tools or technologies. A recent user-centric study [9] found that although the most prominent mashup platforms (e.g., Yahoo! Pipes, Dapper or Intel Mash Maker) simplify the mashup development, they are still difficult to use by non technical users.

Marmite [26] is a tool specifically tailored for integrating and accessing information sources. With respect to other platforms, it offers a more intuitive composition paradigm, which has been devised by means of a user-centered process: it allows users to easily program a set of source filtering operators that can then be connected into a data flow. In line with our approach, Marmite goes in the direction of easing mashup development, for example ensuring continuous feedback through an immediate visualization of the included services and the overall resulting mashup. This works however is still centered on a dataflow paradigm, which in our opinion does not abstract enough from the technical background, requiring for example the users to define operator chaining by means of parameter coupling.

An approach specifically conceived to assist inexperienced end users is proposed in [17], where authors introduce an environment that exploits semantic technologies to relieve users from the complexity of manually resolve intra-service dependencies. This approach has several commonalities with our work. However, it lacks flexibility: the mashup composition is indeed based on templates that in a sense constraint the user choices about service and composition patterns.

Our work tries to overcome the previous limitations, allowing end users to develop their own mashups through an intelligible paradigm that abstracts from technical variables. The aim is to maximize some well-known principles that characterize End User Development [3,4,7,10]. In particular, our approach provides a composition environment that can facilitate the creation of disparate applications accommodating the diversity of the needs, interests and activities that end users want to perform through computer systems. DashMash is indeed a general-purpose mashup environment in which however the risk of becoming too general, thus in some cases ineffective, is limited by the possibility to be customized through the development of ad-hoc components and the registration into the platform of out-of-shelf resources that are of interest to the domain-specific activities that the users need to tackle. In other words, our platform tries to provide the right trade-off between extremely general systems and highly specialized, domain-specific applications that on the other hand cannot be generalized, adapted or evolved [7, 11].

7 Conclusions

In this article, we have proposed our perspective on mashups, mashup tools, and lightweight mashup development processes, arguing that enabling web users (in the consumer context) or employees (in the business context) to develop own applications demands for a high degree of assistance and intelligible concepts. Our proposed approach is a first attempt towards the realization of this objective. However, some more efforts are needed on the following ingredients:

- *Easy-to-use APIs*: Expressive models and description languages for data, application logic, and user interface components are needed to facilitate the component integration within mashups. Suitable discovery and selection facilities (e.g., registries and protocols) are needed as well.
- Design aimed at *interoperability*: Services and mashups should be interoperable, meaning that they must feature cross-platform reusability. Although some proposals exist for mashup-specific standards [20], any mashup platform keeps using its own models and description languages.
- *Dependable mashups*: Although the current efforts are mainly devoted to the improvement of the previous aspects, it is unquestionable that mashups also need to address issues like reliability, transactions, and security – especially if used in business contexts.

DashMash addresses the currently still low ease-of-use of APIs (by definition, APIs are still oriented toward programmers, not end users) and their generally low interoperability (e.g., in terms of supported communication protocols or data formats) by wrapping them and transforming their data into an internal, canonical format that can be understood by other wrappers. This task, however, requires the intervention of expert developers, and cannot be accomplished by the users themselves. As for the dependability of mashups, DashMash does not provide any specific solution, as so far we support non-critical application scenarios only. We have however planned some extensions to address these features.

Finally, while lightweight development processes are needed to alleviate the effort of mashup developers and especially end users, the development of services to be integrated into mashups is a demanding activity, to be performed according to traditional development processes by professional programmers. After all, if on the one hand the success of a mashup is influenced by the added value that the final combination of services is able to provide, on the other hand it is self-evident that the quality of the final combination is strongly influenced by the quality of each single service. Defining models and techniques for developing “good” services and for assessing their quality is therefore another promising direction of our current research, which can give a fundamental contribution towards the development of quality mashups and to aid user innovation [5].

As future work, we aim at exploring different composition solutions, to address, for example, the cooperative definition of mashups (a feature that can greatly enhance team-based cooperation in the enterprise context), as well as an extension of the recommendations mechanisms based on the emergence of composition patterns from the community’s mashups [21]. We also aim at investigating mashup interoperability, for example making DashMash mashups compatible with emergent standards, such as Enterprise Mashup Markup Language (EMML) [20].

References

1. Balasubramaniam, S., Lewis, G.A., Simanta, S., Smith, D.B.: Situated Software: Concepts, Motivation, Technology, and the Future. IEEE Software, 50–55 (November-December 2008)
2. Barbagallo, D., Cappiello, C., Francalanci, C., Matera, M.: A reputation-based DSS: the INTEREST approach. In: Proc. of ENTER 2010 (2010)
3. Bottoni, P., Costabile, M.F., Levialdi, S., Matera, M., Mussio, P.: Principled Design of Visual Languages for Interaction. In: Proc. of VL 2000, pp. 145–155. IEEE CS, Los Alamitos (2000)
4. Burnett, M., Cook, C., Rothermel, G.: End-User Software Engineering. C ACM 47(9), 53–58 (2004)
5. Cappiello, C., Daniel, F., Matera, M.: A Quality Model for Mashup Components. In: Gaedke, M., Grossniklaus, M., Díaz, O. (eds.) ICWE 2009. LNCS, vol. 5648, pp. 236–250. Springer, Heidelberg (2009)
6. Cappiello, C., Matera, M., Picozzi, M., Sprega, G., Barbagallo, D., Francalanci, C.: DashMash: a Mashup Environment for End User Development. Tech. Rep. (March 2011)
7. Costabile, M.F., Mussio, P., Parasiliti Provenza, L., Piccinno, A.: Supporting End Users to Be Co-designers of Their Tools. In: Pipek, V., Rosson, M.B., de Ruyter, B., Wulf, V. (eds.) IS-EUD 2009. LNCS, vol. 5435, pp. 70–85. Springer, Heidelberg (2009)

8. Daniel, F., Casati, F., Benatallah, B., Shan, M.-C.: Hosted Universal Composition: Models, Languages and Infrastructure in MashArt. In: Laender, A.H.F., Castano, S., Dayal, U., Casati, F., de Oliveira, J.P.M. (eds.) ER 2009. LNCS, vol. 5829, pp. 428–443. Springer, Heidelberg (2009)
9. De Angeli, A., Namoun, A., Nestler, T.: End User Requirements for the Composable Web. In: Daniel, F., Facca, F.M. (eds.) ICWE 2010. LNCS, vol. 6385, pp. 396–407. Springer, Heidelberg (2010)
10. Fischer, G.: End-user Development and Meta-Design: Foundations for Cultures of Participation. In: Pipek, V., Rosson, M.B., de Ruyter, B., Wulf, V. (eds.) IS-EUD 2009. LNCS, vol. 5435, pp. 3–14. Springer, Heidelberg (2009)
11. Fischer, G.: Beyond Binary Choices: Understanding and Exploiting Trade-Offs to Enhance Creativity. First Monday 11 (2006)
12. Hughes, T.P.: The evolution of large technological systems. In: Bijker, W.E., Hughes, T.P., Pinch, T.J. (eds.) The Social Construction of Technology Systems: New Directions in the Sociology and History of Technology, pp. 51–82. MIT Press, Cambridge (1987)
13. Hornbæk, K.: Current practice in measuring usability: Challenges to usability studies and research. International Journal of Human-Computer Studies 64(2), 79–102 (2006)
14. Iyer, B., Davenport, T.H.: Reverse Engineering Google's Innovation Machine. Harvard Busines Review 86(4), 58–69
15. Jhingran, A.: Enterprise information mashups: integrating information, simply. In: Proceedings of VLDB 2006, pp. 3–4 (2006)
16. Maula, M., Keil, T., Salmenkaita, J.-P.: Open Innovation in System Innovation Contexts. In: Open Innovation: Researching a New Paradigm, ch. 12, pp. 249–257 (2006)
17. Mehandjiev, N., Lecue, F., Wajid, U., Namoun, A.: Assisted Service Composition for End-Users. In: Proc. of ECOWS 2010, Ayia Napa, Cyprus, pp. 131–138. IEEE CS, Los Alamitos (2010)
18. Obrenovic, Z., Gasevic, D.: Mashing Up Oil and Water: Combining Heterogeneous Service for Diverse Users. IEEE Internet Computing, 56–64 (November/December 2009)
19. Ogrinz, M.: Mashup Patterns: Designs and Examples for the Modern Enterprise. Addison-Wesley, Reading (2009)
20. OMA. EMML Documentation. Technical report, Open Mashup Alliance (December 2010), `http://www.openmashup.org/omadocs/v1.0/index.html`
21. Picozzi, M., Rodolfi, M., Cappiello, C., Matera, M.: Quality-based Recommendations for Mashup Composition. In: Daniel, F., Facca, F.M. (eds.) ICWE 2010. LNCS, vol. 6385, pp. 360–371. Springer, Heidelberg (2010)
22. Roy Chowdhury, S., Rodríguez, C., Daniel, F., Casati, F.: Wisdom-Aware Computing: On the Interactive Recommendation of Composition Knowledge. In: Proceedings of WESOA 2010. Springer, Heidelberg (2010)
23. Sabbouh, M., Higginson, J., Semy, S., Gagne, D.: Web mashup scripting language. In: Proceedings of WWW 2007, pp. 1305–1306 (2007)
24. von Hippel, E.: Democratizing Innovation. MIT Press, Cambridge (2005)
25. M. Weiss, Gangadharan, G.R.: Modeling the Mashup Ecosystem: Structure and Growth. R&D Management (2009) (accepted for publication)
26. Wong, J., Hong, J.I.: Making Mashups with Marmite: towards end-user Programming for the Web. In: Proc. of CHI 2007, pp. 1435–1444 (2007)
27. Yu, J., Benatallah, B., Saint-Paul, R., Casati, F., Daniel, F., Matera, M.: A Framework for Rapid Integration of Presentation Components. In: Proc. of WWW 2007, pp. 923–932 (2007)

Versioning for Mashups – An Exploratory Study

Sandeep Kaur Kuttal, Anita Sarma, Amanda Swearngin, and Gregg Rothermel

Department of Computer Science and Engineering
University of Nebraska-Lincoln, Lincoln, NE 68588, USA
{skuttal,asarma,aswearng,grother}@cse.unl.edu

Abstract. End users with little software background are developing numerous software applications using devices such as spreadsheets, web mashups, and web macros. Web mashups are particularly popular because they are easy to create and there are large public repositories that store them and allow their reuse. Existing repositories, however, provide no functionality for tracking the development histories of mashups. We believe that versioning capabilities can help end users develop, understand, and reuse mashups. To investigate this belief, we created a versioning extension for Yahoo! Pipes – a popular mashup environment – and conducted an exploratory study of users utilizing the environment. Our results show that versioning information allows users to perform mashup creation tasks more correctly and in less time than users not having that information, while also improving the reusability of pipes.

1 Introduction

It is estimated that in 2012 there will be 90 million end-user programmers in American workplaces [14], using devices such as spreadsheets, web mashups, and web macros to automate common tasks. End users such as these create their own code and reuse others' code to achieve a variety of goals. Often, these goals involve combining content from various websites. This has led to the development of programming environments supporting *web mashups*.

Web mashups integrate multiple data sources or APIs into one interface from various web applications. Input can be from different sources and of different formats. Similarly, outputs of mashups can be formatted in different ways and displayed in various media (SMS alerts, email alerts, etc.). Mashups can solve many different problems and can be implemented relatively quickly. They can be easily reused or distributed to other users in the community for reuse or modification, and mashups can be used as components of other mashups.

Mashup programming environments provide central repositories to end users where they can store their mashups. However, current environments do not provide facilities by which users can keep track of the versions or histories of the mashups that they create. In the professional software engineering community, versioning is widely acknowledged as beneficial for activities such as code understanding, change traceability, debugging and maintenance [5]. Versioning systems provide an environment within which previous states of resources (both content

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 25–41, 2011.

and naming) can be easily retrieved. Versioning, therefore, allows engineers to browse through past and alternative versions of a resource and identify how these different versions differ from each other.

We conjecture that the addition of versioning to mashup programming environments may (1) help end users create mashups more efficiently, (2) make mashups more reusable, (3) help users understand the evolution of mashups, and (4) allow users to utilize these advantages without being aware of the underlying complex functions such as check-in, check-out, and so forth.

To investigate this conjecture we are adding support for versioning to a mashup programming environment. We chose Yahoo! Pipes as a platform for this effort. Yahoo! Pipes is a commercial programming environment that allows users to aggregate and mashup content from around the web. The primary reason for selection of Yahoo! Pipes is its popularity; since its launch in February 2007, over 90,000 developers have created individual pipes on the Yahoo! Pipes platform, and pipes are executed over 5,000,000 times each day [12]. The platform also has the advantages of being free, and of utilizing data that can be captured and manipulated by external systems.

In this paper, we report on the first step of our effort to create and study this versioning system. Our approach automatically captures versions without the users' direct involvement, and it allows users to access and navigate these versions through its interface. To explore the potential costs and benefits of using our versioning approach we conducted a "think-aloud" study, in which we observed users attempting to create and understand pipes with and without the aid of versioning information. Our results show that versioning information can help users create pipes more efficiently, and with fewer errors, than when no such information is available.

2 Background and Related Work

Many web mashup programming environments exist, including Yahoo! Pipes [1], JackBe [4], xFruits [7], and IBM Mashup Center [2]. While these environments are more simple than professional programming environments, they are not trivial. There has been recent research aimed at understanding the programming practices and hurdles that mashup programmers face in mashup building. Cao et al. [8,9] discuss problem solving attempts and barriers that end users face while working with mashup environments, and describe a "design-lens methodology" to view programming. Rosson and Zang [18] investigate the types of information that users encounter and the interactions between information sources that they find useful. The authors also examine data gathering and integration [17] and discuss results of a study of web users focusing on their perceptions of what mashups could do for them and how they might be created.

Yahoo! Pipes [1] is arguably the most popular mashup creation environments, and is being used both by professional and end user programmers. Yahoo! Pipes is a visual programming environment in which modules are placed on a canvas and are connected together using pipes. Figure 1 shows the Yahoo! Pipes interface. Yahoo! Pipes consists of various APIs and modules. The input and output

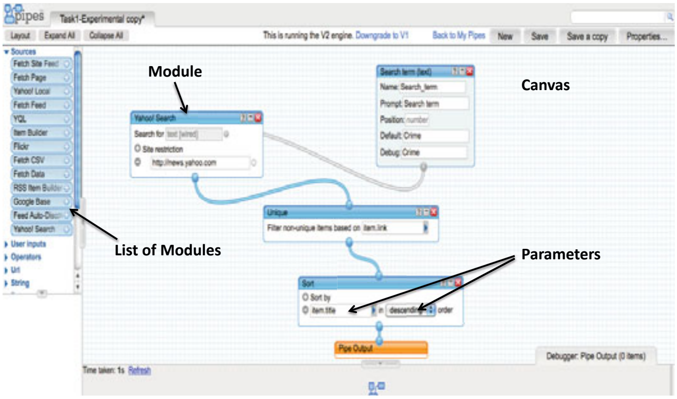


Fig. 1. Yahoo! Pipes environment with a typical pipe

between modules are primarily RSS feed items (e.g., some elements of *item* are date-time, url, location, text and number). RSS feeds consist of item parameters and descriptions. The Yahoo! Pipes modules provide manipulation actions that can be executed on these RSS feed parameters.

Yahoo! Pipes, like other mashup environments, provides a central repository where users can store the mashups they create and use mashups created by others. A user can copy an existing pipe from the repository, modify it to suit one's need, and then publish it to the repository. Such copying of pipes is termed "cloning" and is a common development practice. However, the Yahoo! Pipes server does not allow users to keep versions or histories of these mashups, which makes understanding the development histories of these mashups difficult.

Versioning capabilities are heavily used in commercial software development and are required for a team to be successful. Versioning is largely used by professional developers to keep track of changes (theirs and others), share or benchmark the latest versions of their code, or revert their changes [15]. Some end user environments such as Google Docs and Google Websites provide basic versioning facilities for enabling basic group editing, but these capabilities are only for text edits. Thus far no versioning tools that we know of exist for mashups.

3 Versioning for Yahoo! Pipes

We have developed versioning capabilities for Yahoo! Pipes that allow users to save and navigate between different versions of pipes. Versions are unobtrusively and automatically generated when users save or clone their pipes. This feature is available for pipes that users create themselves as well as for other pipes that they may reuse. Our Yahoo! Pipes extension allows a user to view, edit, or run an older version of a pipe. The versioning tool is operational for most web browsers (Internet Explorer, Firefox, and Safari).

Figure 2 presents our system architecture. We use a proxy server managing communications between the client (web browser) and the Yahoo! Pipes server.

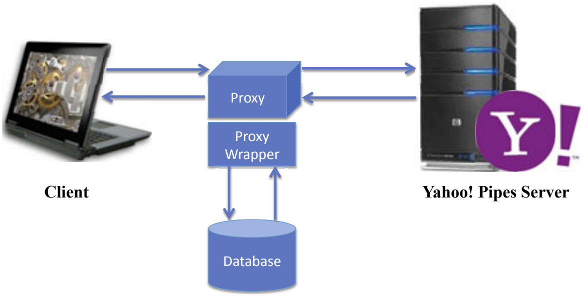


Fig. 2. Architecture of the Versioning System

Using Internet Content Adaptation Protocol (ICAP) [3], we intercept the messages sent between the Yahoo! Pipes server and a client. Our proxy wrapper selects appropriate user events (e.g., save or run) and message contents to create and store versions in a central repository (MySQL database), which serves as the versioning repository. When a user saves his or her pipe, a new version is created for that pipe (e.g., V1, V2, V3,....,Vn). Typically, version control systems allow developers to mark stable or significant versions of the development tree as baselines for easier access and retrieval [11]. When a user *runs* their pipe, we tag that version as a baseline.

The user interface of our extension adds three widgets to the Yahoo! Pipes client interface (see Figure 3). The first two are buttons that allow a user to browse different versions of a pipe via "Undo" or "Redo" operations. "Undo" renders the previous version, while "Redo" renders the next version. Finally, a drop-down list named "History of Pipes" allows a user to select a desired version from the list of available versions for the pipe. This list also displays the modules added or removed per pipe, per version, so that users can view the differences between versions. The foregoing are initial versioning capabilities that we have created. Based on user studies we will determine other appropriate versioning capabilities for end users and ways to intuitively present them.

Note that most versioning systems are text-based and operate at the level of files, whereas we version at the level of modules (Yahoo! Pipes modules). Table 1 presents a comparison of features between our versioning extension to Yahoo! Pipes and typical CM systems.

Our "history list" view provides information about the modules that are added or deleted per version, which is similar to the "diff" feature typically provided by traditional CM tools. We do not yet provide "merge" functionality whereby two or more changes in the same or different modules can be merged. To help end users better understand and navigate across versions we provide the undo and redo features, which matches the revert functionality provided by CM systems. We believe that it will be easier for end users to learn and adopt the undo-redo features to navigate across pipe versions, as these functionalities are common in most text editors. Finally, we mimic tagging of stable versions in CM systems as baselines. That is, when a pipe is *run*, we treat it as a version that has been completed and is in a stable form and mark it as a baseline in the history of the pipe.

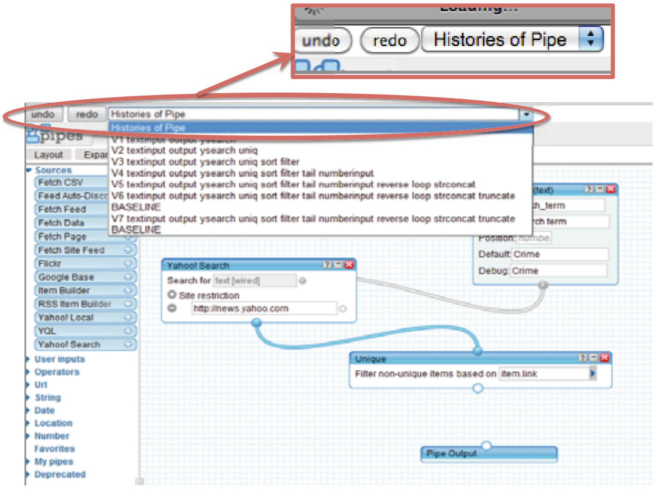


Fig. 3. Yahoo! Pipes extension interface where version V2 of a pipe is displayed. The list view (History of Pipe) shows seven existing versions and a few baseline tags.

Table 1. Comparison of the features implemented in the versioning extension for Yahoo! Pipes with existing configuration management systems

Features implemented	CM Systems	Versioning extension in Yahoo! Pipes
Versioning unit	File level	Module level
Versions created	On commit	On save/cloning of pipe
Browsing	Undo or selecting version	Undo, redo or selecting from list view
Snapshots	Baseline	Run
Diff	File level	Module level
Deltas	Add, Delete and Modify	Add and Delete
Merge	Implemented	Future work
Target population	Professional programmers	End-users

4 Empirical Study

To investigate the effects of building versioning capabilities into a mashup programming environment we conducted a user study, considering the following research questions:

RQ1: Does versioning allow mashup programmers to efficiently and effectively perform tasks related to mashup creation?

RQ2: Does versioning help mashup programmers understand complex third-party mashups?

4.1 Participants

While ultimately our interest in mashups involves end users, including those who do not have experience programming in more formal languages, for this first, exploratory study, we believed it would be better to choose participants who were readily available and had some programming experience in environments other than Yahoo! Pipes. This would allow us to make sure that participants were able to quickly grasp the programming concepts in Yahoo! Pipes through tutorials that could be provided as part of a study. Another reason for selecting users with basic computer knowledge as participants was to ensure that they would be able to work with sufficiently complex pipes, since benefits of versioning are typically evident only for software artifacts that are non-trivial. Past work has also shown that complete novices to programming found coding in Yahoo! Pipes to be non-trivial and needed external help, even when pipes were three modules in size [18]. We thus focused on students who were computer science or computer engineering majors, or who at least had taken one programming class.

To recruit students we sent an email to a departmental mailing list. As an incentive participants were included in a raffle for a $25 prize. Nine students responded to our advertisement. All students were male, with seven from computer science or computer engineering and two from other departments. Four of the students were undergraduates and five were graduates. None had any prior experience with Yahoo! Pipes, but all had some programming knowledge (78% knew multiple programming languages).

4.2 Study Setup and Design

The study used a single factor, within-subjects design. (A within study design [16] is one in which the independent variable is tested with each participant.) The independent variable was our versioning extension to Yahoo! Pipes; that is, participants in the control group completed their tasks without our versioning extension and participants in the experimental group used our extension. We opted for a within-subject design for two reasons. First, we wanted to minimize the effects of individual differences among participants, which can be reduced through a within-subjects study design. Second, a within-subject study afforded us more data using a smaller sample size, which was more appropriate for our exploratory study and the use of a think-aloud protocol (see below). Finally, since each participant in the within-subjects study gained experience in performing tasks using the environment with and without our versioning extension, they were better positioned to give feedback about the usefulness and usability of our versioning extension than participants in a between-subject study.

We used think-aloud protocol in our study, where participants were asked to vocalize their thought processes and feelings as they performed their tasks [13]. We used think-aloud protocol because a primary objective of this exploratory study was to gain insights into the users' thought processes – barriers and problems that a user faced when using Yahoo! Pipes and our extension. This approach required us to administer the study to participants on an individual basis with an observer – in this case, the first author.

We performed the user studies in the Usability Lab of the Computer Science Department at the University of Nebraska-Lincoln. At the beginning of the study, participants were asked to complete a brief background questionnaire, which was followed by a brief tutorial of approximately ten minutes on Yahoo! Pipes and versioning in general. The tutorial also included a short video of a sample think-aloud study so that users could understand the process.

After completion of the tutorial, we asked participants to create a small sample pipe to give them hands-on training in creating a pipe and familiarity with Yahoo! Pipes. We also provided them with a list of external web sites where they could find further help with the environment, some of which were within the Yahoo! Pipes environment and some of which were external web pages.

Following these preliminaries, participants were asked to complete tasks for the study. Each participant completed a pair of tasks for RQ1 followed by a pair of tasks for RQ2. The first of each pair of tasks was a task without versioning support (control tasks), and the second was a task with versioning support (experimental tasks). We audio recorded each session and logged the users' on-screen interactions using a screen capture system (Morae [6]). The total times required for completion of the study per participant was approximately 1.5 hours, which included on an average of 60 minutes for task completion.

After all tasks were completed, we administered a survey to obtain feedback and additional thoughts from participants. The survey consisted of both closed and open-ended questions about the tasks, the interface of the versioning extension, and the experimental process.

4.3 Tasks

Our analysis of large numbers of Yahoo! Pipes showed that users typically create pipes containing between five and 70 modules, and encompassing a broad spectrum of complexity. It is likely that versioning capabilities will not be useful for simple pipes; however, our belief is that they can be useful in understanding and building upon complex pipes. We therefore created tasks with levels of complexity that we believed were sufficient to investigate this belief.

Specifically, we defined two types of tasks, Task 1 and Task 2, addressing our research questions on whether versioning helps users create pipes of their own, and whether versioning helps users in understanding complex, third-party pipes. Because the study was a within-subjects study, we further subdivided each task into two categories: control (C) and experimental (E) tasks. Therefore, in total we defined four tasks with Task1.C and Task1.E addressing RQ1, and Task2.C and Task2.E addressing RQ2, respectively. For each of these tasks, we created an appropriately sized Yahoo! Pipes example for participants to utilize.

Task 1 largely required a user to create a relatively small pipe as per a given set of requirements. However, users were given a similar existing pipe (of ten modules) as a template that they needed to first understand. A large portion of the sample pipe could be reused in implementing the pipe required by the task. The existing pipe for Task1.C allowed a Spanish-speaking person to search for reviews of any business (e.g., museum, university), within a geographical location (e.g., San-Francisco), and within a certain distance (e.g., 200 miles), the first two being provided by the user and the last being hard coded into the pipe. The pipe user could also choose the number of reviews to view. All results returned contained the search term in the title, and were sorted in alphabetical order. Participants were asked to review the pipe until they understood its modules and linkages and then move onto the second part of the task.

The second part of Task1.C required the participant to build a pipe that used a subset (three to four) of the modules from the sample pipe that they had just reviewed, while implementing additional functionality requiring two to three other modules. In particular, the user was asked to create a pipe that allowed a search for a review of any item around any area within a certain specified distance, but required an additional functionality of changing the original titles of the search results to titles of their choice.

Task1.E involved a different pipe of complexity similar to the one used in Task1.C. This pipe was designed for a news enthusiast who wanted to search Yahoo! News for a topic specified by the user, filtered to display only those news items with the keyword "crime" in the title. The results of the query were to be unique and contain at least two items in reverse order based on the date of publication. As in Task1.C, the participants were asked to understand the given pipe and then move on to the next step. In the second part of Task1.E, the user was asked to create a pipe that allowed a French user to search Yahoo! News for a search term while making sure that search result titles were unique and translated into French. Note that the pipe used in this task had an associated versioning history that was accessible to the user.

Task 2 required the user to view a given pipe and answer a set of multiple-choice questions about the pipe and its functionality. The pipes used in Task 2 (Task2.C, Task2.E) were larger and more complex than those used in Task 1. Task2.C involved a pipe of 47 modules that displayed a list of unique, "mashed up" contents from five different sites specified by the user. The user could also limit the number of results that were displayed and sort results in descending order of date. Task2.E involved a pipe of 50 modules and was actually a generic filter to merge feeds from different sources, remove duplicates and ensure unique items in those feeds. Further, a user could specify four different feeds, truncate or limit the maximum number of resulting items per feed, and select a maximum number of items that could be displayed. Note that a versioning history for this pipe was available so that participants could investigate how the pipe was built from the ground up to better understand the different functionalities provided by the modules in the pipe.

4.4 Threats to Validity

The primary threat to external validity in our study is that all our participants were male and seven out of nine were Computer Science or Computer engineering majors. Computer Science and Engineering majors are not representative of other classes of end users who use Yahoo! Pipes. Still, they do represent one class of users and a class that could conveniently be studied initially. Given the lessons learned in this study, we can proceed to create a larger-scale controlled experiment on different classes of users. A second threat to external validity is that our study considered only two tasks that built on only two types of pipes. Additional studies are needed to examine other tasks and other types of pipes.

Where internal validity is concerned there are several threats to consider. The primary threat to this particular study relates to our choice of a within-subjects design. This study design helped to minimize the effects of individual differences and the use of a small pool of participants, but it might have led to learning effects as the participants moved from initial to later tasks. Another threat is that our tasks for control and experimental groups were not counter-balanced; this could have led to a bias in the performance of the experimental tasks. A further threat could occur if the control and experimental tasks were not comparable in terms of complexity; however, our data shows that participants required similar times to understand pipes in the control and experimental tasks, suggesting that the pipes were of similar complexity.

Threats to construct validity include the possibility that the complexity of our pipes was not high enough to allow measurements of effects, and that the pipes used for control and experimental tasks were not comparable in complexity. We controlled for this by performing initial pilot studies on non-participants and using their feedback to adjust the pipes and the tasks.

4.5 Results and Analysis

Research Question 1

To address our first research question we frame two separate hypotheses, involving the time required to create pipes with and without versioning and the correctness of the pipes thus created. We consider each hypothesis in turn.

Ha1.1: Less time is required to create a pipe when using versioning than when versioning support is absent.

As participants performed each task relevant to RQ1 (Task1.C and Task1.E), we examined the total times required by participants to complete their tasks, which included both the times to understand the given sample pipe and to implement the required pipe. Figure 4 shows these times as proportions of the total time for Task1.C and Task1.E – times spent on understanding and implementing pipes. The average time spent by all participants in Task1.C was 17.19 minutes (median 14.28) while the average time spent for Task1.E was 7.22 minutes (median 6.68). We used the non parametric Wilcoxon signed-rank test to test the difference in measurements between the two experiment alternatives on each sample. The test yielded W=43 with p-value = 0.006 and confirmed the hypothesis, so we conclude that it took participants more time to complete Task1.C than to complete Task1.E.

We wished to further investigate the benefits of versioning support with respect to the subtasks of the pipe creation: namely, understanding and implementing the pipe. We did this by individually analyzing the times used by participants in these tasks (having measured each individually during the study).

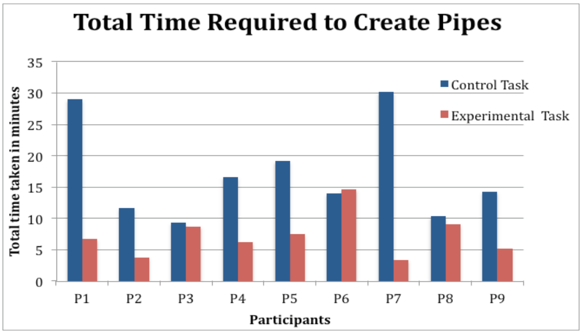


Fig. 4. Total time spent to create a pipe for a given task

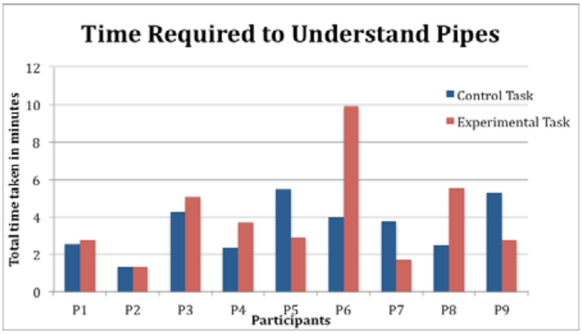


Fig. 5. Time required to understand pipes

Figure 5 shows the time required by participants to *understand* the pipes they were given in Task1.C and Task1.E. The average time taken by users to understand the pipe in the control task was 3.49 minutes (median 3.72), while in the experimental task it was 3.94 minutes (median 2.88). As Figure 5 shows, differences in the time required to understand the pipes varied in magnitude and across tasks. This means that versioning support did not appear to provide any significant advantage in the understanding task. We conducted the Wilcoxon Signed-rank test at a 95% confidence interval on the data, and the p-value in this case was 0.363 (with W = 15); thus we cannot assert that there were any differences in understanding the pipes during Task1.C and Task1.E.

A post-hoc analysis of participant behavior showed that they typically did not use the versioning features while engaged in the understanding task even when these were available, so it is difficult to ascribe the lack of differences in understanding time across tasks to the presence or absence of versioning.

We next turned to the *implementation* task. Figure 6 shows the time required to implement pipes for each participant. As the figure shows, for all participants the time required to implement pipes was less in the experimental task than in the control task. The average time required to create pipes in the control task

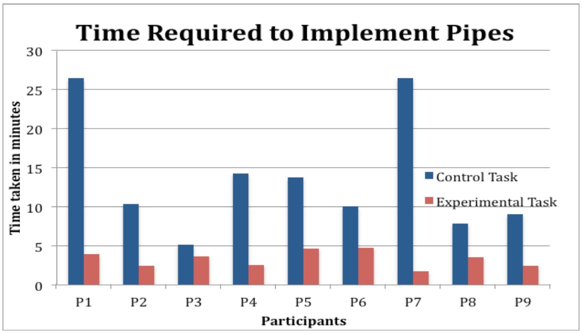


Fig. 6. Time required to implement pipes

was 13.70 minutes (median 10.36), while the average time required to create pipes in the experimental task was 3.27 minutes (median 3.52). A Wilcoxon Signed-rank test had W=45 with p-value = 0.002. We therefore confirmed that the differences in times were statistically significant. It would seem, then, that the observed differences in overall pipe creation time stem from differences in implementation times.

Conceivably, one cause of the foregoing differences may be learning effects; that is, participants may have grown sufficiently familiar with the pipes during the control task to allow them to act more quickly in the experimental task. However, we noted during our observations that most of the time spent in the control task involved users examining the modules of the pipe that they had cloned to determine their applicability for the new pipe. In the experimental task, in contrast, participants were able to select earlier versions with only the modules that were required and thus save time that would otherwise be spent examining and removing the extra modules. The time differences related to the pipe implementation thus do seem to be ascribable to versioning.

Ha1.2: Pipes that are created with versioning support are more correct than those that are created without versioning.

As our participants attempted Task1.C and Task1.E, we allowed them to proceed to the next task when they believed that they had finished creating the pipe as per the requirements and as correct as possible. Since there was no acceptance test, some of the pipes were not completely correct.

To allow us to assess correctness, prior to the experiment the first and third authors created solutions to the tasks; these served as oracles for grading. Following the study, we measured the correctness of pipes by assigning each module in the pipe a certain weight which was then aggregated. More specifically, the weight depended on the number of parameters that the module required as well as the module's existence, with each element being equally weighted. For example, the *Sort* module has two parameters and was a required component for the pipe. We would, therefore, assign 1/3 point if the module was included and 1/3

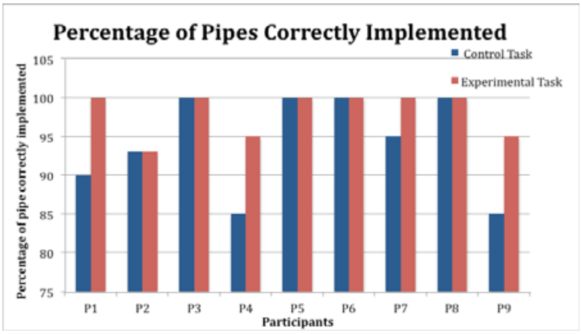


Fig. 7. Correctness of pipes created by participants

point to each correct parameter making the total points for the module to be 1. We deducted 5% from the total points given to the pipe if extra modules were included that affected the output. If a required module was completely missing, we deducted the points that would have otherwise been assigned for that module (1 point in the case of *Sort*).

The foregoing scoring procedure was conducted independently by the first and third authors, and then the scorers met to check the results of their grading and ensure that no grading errors had been made.

Figure 7 presents the percentages of correct pipes that were created by participants in the first task. Participants were largely successful in creating correct pipes in both the control and experimental tasks, with an average correctness for the experimental task being 98.1% and an average correctness for the control task being 94.2%. As the figure shows, four participants created completely correct pipes. Participants in the experimental tasks did exhibit a slightly higher measure for correctness of pipes than when created in the control task. We used a Wilcoxon Signed-rank test at a 95% confidence level giving us a W=0 and a p-value of 0.045; hence, we conclude that the differences in correctness between the two groups are statistically significant, confirming hypothesis Ha1.2.

Research Question 2

Task 2 was designed to help us address our second research question regarding the benefits of versioning in enabling the understanding of complex, third-party mashups. We investigate this question through our hypothesis:

Ha2: Versioning will help users learn about the mashups they encounter.

In addition to observing user behavior in Task 2, we administered a multiple choice quiz in which participants were asked to answer questions to help us assess the degree to which they understood the pipes. Figure 8 summarizes the results of the questionnaire, showing that the participants differed little in terms of the numbers of correct and incorrect answers in both the control and experimental

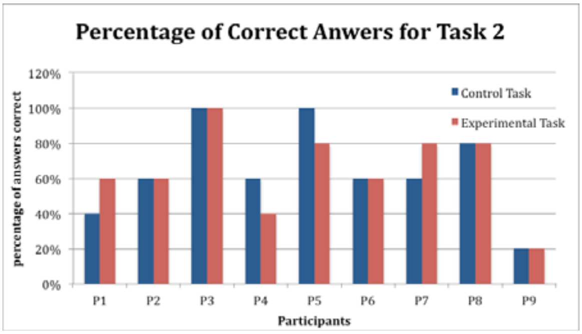


Fig. 8. The percentage of correct answers about pipes given by users in quizzes associated with Task 2

tasks. We evaluated these results through the Wilcoxon Signed-rank test at a 95% confidence interval with W equal to 14.5. Our p-value of 0.674 indicates that no significant difference in understanding was observed for tasks with and without versioning support. In our further analysis of participant behavior, we noticed that a majority of users (five of nine) did not refer to the versions when performing Task2.E, which is similar to our observations in Task1.E.

Despite these results, we note that in the exit survey seven out of nine participants stated that they found versioning helpful for learning what a pipe did. It is possible that the Task 2 quiz result is more reflective of the difficulty in measuring a participants' ability to understand a particular pipe than of the participants use of versioning to do so.

To better understand how versioning helped or could have helped users we further investigated participant behavior in Task 1 since a significant component of the task involved understanding a pipe. We largely depended on experiment notes and the transcripts of the think-aloud sessions to do so. We found that participants, especially those performing control tasks, often desired to return to the original version of the pipe to understand it better. In some other cases participants had deleted more modules than required and wished to bring them back. In both situations a versioning tool could have been helpful.

Further, in Task 1 several participants struggled when they tried to revert to the original pipe to view it and learn how it worked. Participants in Task1.C either cloned their pipe (seven of nine) or constructed it from scratch (two of nine), but in either case they had to refer to the original pipe to look at the modules or its pipe structure. Some participants (P1, P4 and P7) viewed the original pipe by manually opening it in a separate window. With the availability of versioning in Task1.E, however, none of the participants had this trouble, because the versioning system provided access to the original pipe. In Task1.E, in fact, five of nine participants investigated the pipe structure in different versions before deciding which one to use as the base for constructing their pipe. All these participants stated in the exit survey that versioning would help improve the understanding of a pipe.

Finally, we noted that participant P6 in Task1.C also tried to go back to the original pipe, but was unable to locate it from the list of pipes from the Yahoo! page. If this pipe had a versioning tool available then he could have saved the example pipe as a version so he would be able to go back to it whenever he needed to. Note that during the experimental task this participant used versioning to learn about the pipe's structure. He commented that: *"It is easy to look at a partial part of the pipe in the versioning rather than looking at the entire pipe. It is easy to go back to the previous versions."*

This thought was echoed by another participant, P5, who stated that versioning was helpful when trying to learn about the functionality of a pipe: *"When looking at the overall finished pipe it was quite intimidating. Using the versioning tool it was much easier to follow along the order of the finished pipe to gain a better understanding."*

In summary, we can infer that participants used versioning to understand the functionalities of the pipes provided to them to an extent, but we were not able to see any statistically significant difference in the time it took to understand a pipe in Task 1 or the number of correct answers in Task 2. However, our own observations and exit survey suggest that versioning can enhance the understanding of pipes.

5 Discussion

Quantitative analysis of the data confirmed that versioning capabilities can help mashup programmers complete their tasks efficiently and correctly. We now discuss additional findings from our analysis of the experiment data.

One of our initial conjectures about the potential usefulness of versioning for mashups was that it would enable mashup programmers to better reuse mashups, by making earlier versions available. Since reuse is one of the primary mechanisms by which end-user programmers construct new programs [10], data on this conjecture might further inform the mechanisms by which we provide versioning. Our exploratory study indicates that versioning can facilitate the reuse of pipes and their sub-parts. Our results divided the total time taken to create a pipe into two subsets: (1) the time taken to understand the given example pipe and (2) the time taken to implement the required pipe. We found no statistical benefits of versioning for the first of these subsets, but we found that users were quicker in implementing their pipes when using versioning capabilities. Further, they produced more correct pipes. We observed that the benefits of versioning occurred largely because participants used the "versioning history" functionality and were able to correctly use the modules from the example pipe. Our results show that versioning can indeed facilitate the reuse of pipes and their sub-parts.

This benefit of versioning was suggested in the feedback from our exit surveys where we asked participants about their thoughts on whether versioning improved/can improve the reusability of pipes. Eight of nine participants mentioned that they thought versioning did help improve the reusability of pipes and all eight of them had used versioning in their experimental tasks. In fact, one participant commented on the benefits of understanding the thought process of the creator of the pipe as evidenced through the versioning history, which helps in the understanding of the pipe. Participant P4 commented: *"You can see how each pipe was developed along the way to final product."*

Our qualitative analysis of the transcripts of the think-aloud sessions helped us investigate the different types of problems that participants faced when performing their tasks. Here we list some of the instances of typical problems faced by users where versioning could have been helpful. For example, one of the problems was in locating the original pipe (when a pipe was created from scratch) or the parent pipe (when a pipe was cloned for the task) for reference during the tasks. Recall that participant P6 was unable to locate the original pipe, which left him frustrated. Similarly, participants who cloned the original pipe often removed too many modules or did not remember how the modules that they had removed functioned in the original pipe. As an example, participant P7 faced this problem and was frustrated when given the assignment of replacing all of the item titles (Task1.C) with a user input. He was not sure whether a *loop* was needed and he ended up removing the loop and adding it back to the canvas at least three times. Currently, there is no mechanism for identifying the parent pipe in the Yahoo! Pipes environment. Moreover, one can only recognize that the current pipe has been cloned, but there is no way of identifying which modules belonged to the original version and which are new additions.

Versioning capabilities would have helped in both these kinds of problems. In the first case, versioning would have allowed the users to easily identify the original (base) version and build from there. In the second case, our versioning features of "undo" and "redo" would have helped participant P7 to easily explore the functionality of the *loop* module.

We found that versioning histories as provided by our list view were frequently used by the participants when they edited pipes in the experimental tasks. The list view enabled the user to directly select a version of the pipe that they desired to modify. As discussed earlier, participants in Task1.E used the list view to select the version of interest, which they then used to build upon. Participant P3 commented: *"Having versioning helps to see small building blocks makes it easy to see the entire picture. In the future, modifying/editing having all versions will be helpful."*

We were surprised to note, however, that participants did not really use versioning when they were performing the "understanding" activities or for investigating the effects of changes by going backward ("undo") or forward ("redo"). Our initial assumption was that versioning would be helpful in understanding changes – the development process of another user as visible through the versioning histories, or the participant's own development process through undo and redo. None of the participants used these features. This might be because these were novel but unfamiliar functionalities, which users were uncomfortable in using. We plan to explore different kinds of visual metaphors that make such functionality more intuitive and usable to end users.

6 Conclusion

We have presented a versioning tool for use on mashups, and provided insights into the potential usefulness of that tool by conducting a think-aloud study. The results of the study confirmed that the versioning tool can help mashup programmers create mashups more efficiently and effectively. We also see indications that it can improve reusability of code.

In our future work, we plan to implement a more robust "diff" functionality in our versioning extension for Yahoo! Pipes. Currently, the "diff" of pipes is at the module level; that is, it identifies modules that have been added or removed. We also plan to fine-tune our analysis to identify modules that have changed, specifically tracking changes to the parameters in the module and connections across modules. We will also consider providing a side-by-side comparison of two versions of a pipe. Such an interface that also highlights the changed modules should allow users to very quickly identify how a pipe has changed. Finally, we will expand the scope of our studies of versioning to include additional groups of users, and in particular, end users.

Acknowledgments

This work was supported in part by the AFOSR through award FA9550-09-1-0129 and by the NSF through award NSF CCF-0747009 to the University of Nebraska - Lincoln.

References

1. http://pipes.yahoo.com/pipes/
2. http://www-01.ibm.com/software/info/mashup-center/
3. http://www.icap-forum.org/
4. http://www.jackbe.com/
5. http://www.open.collab.net/news/press/2007/svn_momentum.html
6. http://www.techsmith.com/morae.asp
7. http://www.xfruits.com/
8. Cao, J., Rector, K., Park, T.H., Fleming, S.D., Burnett, M., Wiedenbeck, S.: A debugging perspective on end-user mashup programming. In: VLHCC, pp. 149–156 (September 2010)
9. Cao, J., Riche, Y., Wiedenbeck, S., Burnett, M., Grigoreanu, V.: End-user mashup programming: Through the design lens. In: CHFCS, pp. 1009–1018 (April 2010)
10. Cypher, A., Dontcheva, M., Lau, T., Nichols, J.: No Code Required: Giving Users Tools to Transform the Web. Morgan Kaufmann, San Francisco (2010)
11. Estublier, J., Leblang, D., Hoek, A.v.d., Conradi, R., Clemm, G., Tichy, W., Wiborg-Weber, D.: Impact of software engineering research on the practice of software configuration management. TOSEM 14, 383–430 (2005)
12. Jones, M., Churchill, E.: Conversations in developer communities: A preliminary analysis of the Yahoo! Pipes community. In: CCT, pp. 51–60 (June 2009)
13. Lewis, C.H.: Using the "thinking aloud" method in cognitive interface design. RC 9265, IBM (1982)

14. Scaffidi, C., Shaw, M., Myers, B.: Estimating the numbers of end users and end user programmers. In: VLHCC, pp. 207–214 (September 2005)
15. Tichy, W.F.: RCS-A system for version control. In: SPE, pp. 637–654 (1985)
16. Wohlin, C., Runeson, P., Hšst, M., Ohlsson, M., Regnell, B., WesslŽn, A.: Experimentation in Software Engineering: An Introduction. Springer, Heidelberg (2000)
17. Zang, N., Rosson, M.: Whats in a mashup? And why? Studying the perceptions of web-active end users. In: VLHCC, pp. 31–38 (September 2008)
18. Zang, N., Rosson, M.: Playing with information: How end users think about and integrate dynamic data. In: VLHCC, pp. 85–92 (September 2009)

Creating Mashups by Direct Manipulation of Existing Web Applications

Giuseppe Ghiani, Fabio Paternò, and Lucio Davide Spano

CNR-ISTI, HIIS Laboratory, Via Moruzzi 1
56124 Pisa, Italy
{giuseppe.ghiani,fabio.paterno,lucio.davide.spano}@isti.cnr.it

Abstract. We present an environment to enable people without programming knowledge to create mashups composed of Web components selected directly from existing Web applications. The authoring environment allows the creation of communication among components originally belonging to different applications. We report on some example application, the underlying architecture of the environment, and a first user test.

Keywords: Mashups, Web applications, End User Development.

1 Introduction

The Web is still the most common user interface. Millions of Web sites are available and ever more people access them daily for their activities. However, each user has slightly different needs and activities to carry out. This has pushed interest in mashups, whose purpose is to compose user interfaces, contents, and functionalities from various sources. Indeed, Web mashups are Web applications generated by combining content, presentation, or application functionality from disparate Web sources. They aim to combine these sources to create new useful applications. Early studies [12] of Web developers and their experiences with building mashups found that the majority of developers created map mashups, with some creating photo mashups. This has changed over time and environments such as iGoogle have shown that mashups can be used for building a variety of composite applications. These studies also pointed out a number of issues with mashup environments: frustration when dealing with application programming interfaces, documentation, and coding details.

While a number of commercial and research environments to support the development and use of mashups exist, we note that they usually require specific tools and languages for their application or they have limited applicability. Even approaches such as Yahoo Pipes[1], which provides graphical mashup platforms aiming for a partially assisted Web development for less skilled programmers, still requires some effort and technical knowledge in order to be applied. Our goal is to overcome such limitations through an End User Development (EUD) [6] environment for Web mashups, which allows even people who have not software development as their primary

[1] http://pipes.yahoo.com

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 42–52, 2011.

task to create their novel, composite Web applications. For this purpose we propose an environment that does not require the use of specific languages or the development of any specific model beforehand in order to enable the inclusion of parts of a Web application in the mashup. To this end, we simply require the use of a proxy/mashup server for the access to the Web applications that should be included in the mashup. The purpose of this server is to inject some scripts to enable the selection of the components that will be part of the newly created mashup. The authoring environment allows the creation of communication among components originally belonging to different applications. An example that we describe in detail in the paper is to take the search form from Amazon and connect it to multiple services in addition to those from Amazon (e.g. eBay, ...) so that one request can be sent simultaneously to all of them. Our environments aims to support the composition of the various types of components in Web applications: data, application logic, and user interface [11]. The user interface components maintain the original set of event handlers and are composed in terms of layout, data, and application logic (e.g. Web services). In order to obtain the new mashups no specific browser or plug-in is required, the involved applications have only to pass through a specific server. It is only needed the use of browsers able to support DOM serialization (e.g. Internet Explorer 9, Firefox 3.4 or higher, Chrome). The Mashup Enviroment is simply accessible through its Web address.

In the paper, after discussion of related work, we provide an example use scenario that shows the possibilities of our environment. Then, we describe the underlying architecture that makes such scenarios possible, we report on a first user test and, lastly, we draw some conclusions with indications for future work.

2 Related Work

d.mix [3] shares with our approach the use of a proxy server. In d.mix users select marked elements that they wish to copy. Through a site-to-service map d.mix composes Web service calls that yield results corresponding to the user's selection. This code is copied to a wiki for editing and hosting. Thus, d.mix still requires good programming knowledge for editing such code. A tool for rapid development of composite applications using annotated Web services [2] has been one of the tools developed in the ServFace project. This tool aims to support end user developers to build the front-end of existing services starting with the WSDL service operation descriptions, and exploiting Web services annotations providing suggestions for user interface development, when available. However, it is limited to create simple form-based user interfaces, and still requires some familiarity with Web services technology.

Collapse-to-zoom [1] is a technique for viewing Web pages on small screen devices by interactively removing irrelevant content selected through end user gestures performed through a pen on the mobile device screen. Differently, in our case we ask users to select the relevant content from different applications in order to build the mashup.

The existence of a tremendous amount of Web content, which is not always in a form that supports end users' needs has motivated Marmite [13], which allows users to select some operators, place them in a data flow representation, and view the current state of the data at a particular operator in a table, which shows what the data looks like after it has passed through an operator. However, Marmite is able to manage only a small set of pre-defined data types. In addition, it has been implemented as a Firefox plug-in, thus limiting its applicability to only this browser, while we propose a solution that is browser independent thanks to the use of an intermediate proxy/mashup server. Lin and others [5] have proposed a system using direct manipulation and programming-by-demonstration techniques to automatically populate tables with information collected from various Web sites. They have addressed a class of ad hoc mashups, where users want to quickly combine data from multiple Web sites in an incremental, exploratory fashion, often in support of a one-time or infrequently performed task. Their tool allow collecting a data table from anywhere and then running scripts that add columns to that table based on the data in each row while our tool allows users to combine functionalities from different Web sites and build new ones as well.

Some tools have been developed with the purpose of extracting code from Web pages but not with the goal of obtaining dynamic mashups. For example, Firecrow [7] aims to extract the code responsible for the desired behaviour of the selected Web UI control, but it is limited to extraction of the basic standard controls of Web applications. In the context of Web engineering, there also exist two related approaches and tools that facilitate the understanding of dynamic web page behaviour: Script InSight [4] and FireCrystal [8], which have a similar functionality of relating elements in the browser with the code responsible for them. We take this approach further since we enable including the components selected through their user interface directly in the mashup without having to deal with the underlying code.

3 An Example Application Scenario

In order to introduce our EUD mashup environment we describe an example application involving well-known applications.

In the scenario the user accesses Amazon through our proxy/mashup server and starts to search for books on funny t-shirts. After a while it occurs to the user that he may get better results through another application, such as eBay. Thus, he accesses eBay through the same intermediate server. Then, he notices that sometimes Amazon provides better results but in other cases eBay is more useful with the items shown. Thus, the user starts to think that it would be nice to have one single application able to show both results with a single query in order to speed-up the search. For this purpose he selects and sends the following components to the mashup editor: the Amazon search form, a result sample from Amazon and a result sample from eBay. The mashup editor asks the user to name and locate the widgets containing each of the three components selected and the display size associated with them in the mashup layout. Figure 1 shows the environment during this editing work. It is composed of three main elements: on the top-left part there is the mashup editor, which includes the parts selected and copied into it; on the right there is the eBay application highlighting in green the components that have been selected in the page presenting the search results; on the left-bottom there is the Amazon application in which the selected components (in this case the input form for the search) are highlighted.

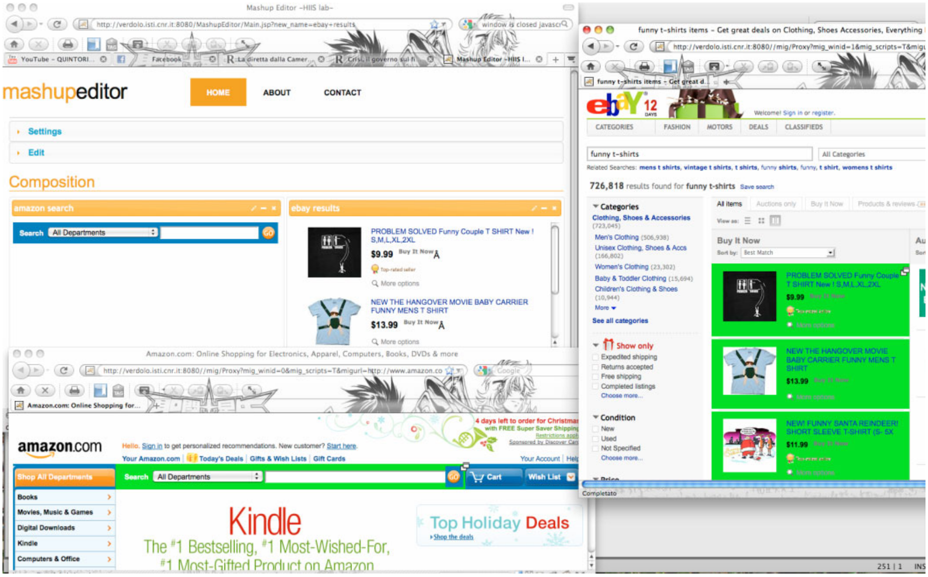
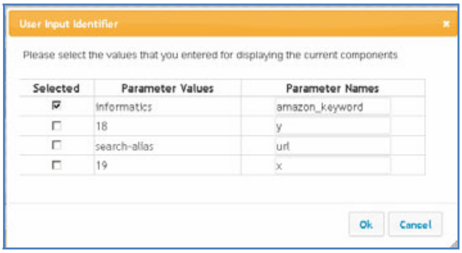


Fig. 1. The Mashup Editor and some Web components interactively selected

Afterwards, the user needs to create the connection between the input from the Amazon form and the eBay and Amazon results. To enable this connection the user has first to select each component presenting the results of a previous request to a Web server. For this purpose the mashup editor shows the list of parameters that have been initially used for generating the results (Figure 2 shows the case for Amazon).



User Input Identifier

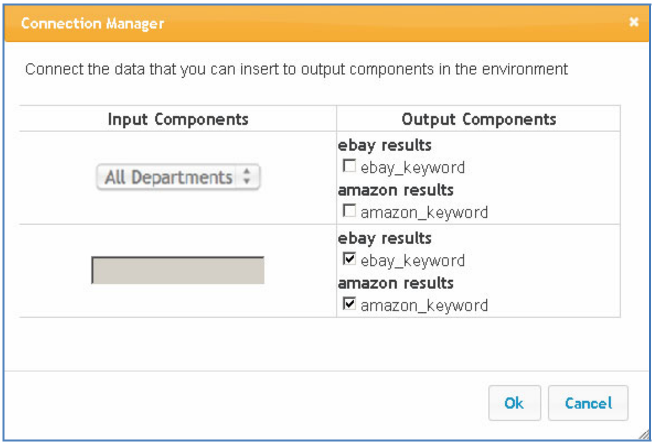
Please select the values that you entered for displaying the current components

Selected	Parameter Values	Parameter Names
☑	informatics	amazon_keyword
□	18	y
□	search-alias	url
□	19	x

Fig. 2. The Dialog Box for selecting the parameter previously entered through an input field

The user only has to indicate which of them has been entered by himself through an input field. This information is useful for identifying a potential connection between this component and other components that provide user-generated input.

Then, the user activates the connections editor, which asks what input and output components to connect (see figure 3). In particular, the connection manager shows on the left-hand side the list of input elements that belong to the selected components, and, for each of them, on the right–hand side the output components that can be connected to a user input element. In our example the user selects the input text field of the Amazon form and then the Amazon and the eBay outputs resulting from a number of parameters.



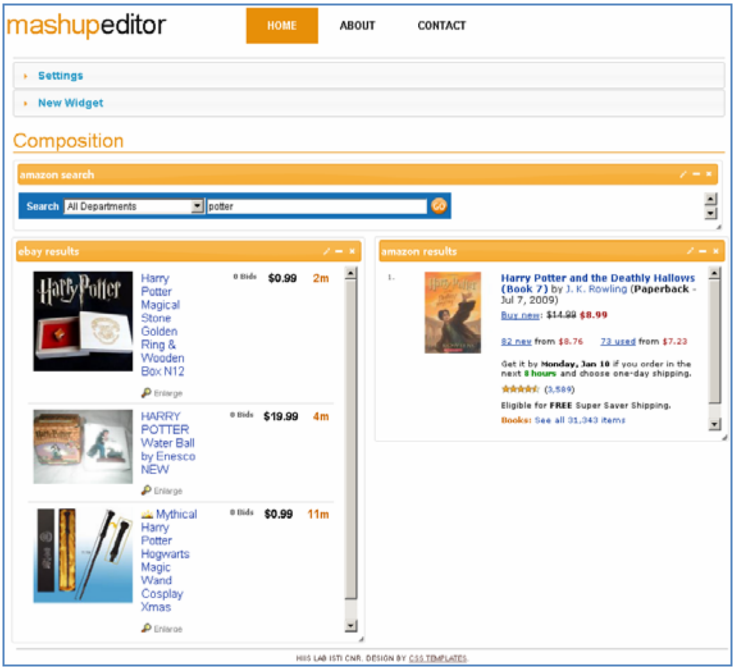
Connection Manager

Connect the data that you can insert to output components in the environment

Input Components	Output Components
All Departments	ebay results □ ebay_keyword amazon results □ amazon_keyword
	ebay results ☑ ebay_keyword amazon results ☑ amazon_keyword

Fig. 3. The Dialog Box allowing the user to associate input elements with output components

Figure 4 shows the resulting mashup working: in this case the user has entered the "potter" term and the results from both Amazon and eBay are shown. Please note that since only one result from Amazon and three from eBay were selected then the mashup shows only the first result from Amazon and the first three from eBay also in the further queries entered by the user.



Search: All Departments, potter

ebay results

Harry Potter Magical Stone Golden Ring & Wooden Box N12	0 Bids	$0.99	2m
HARRY POTTER Water Ball by Enesco NEW	0 Bids	$19.99	4m
Mythical Harry Potter Hogwarts Magic Wand Cosplay Xmas	0 Bids	$0.99	11m

amazon results

1. **Harry Potter and the Deathly Hallows (Book 7)** by J. K. Rowling (**Paperback** - Jul 7, 2009)

Buy new: ~~$14.99~~ **$8.99**

82 new from $8.76 73 used from $7.23

Get it by **Monday, Jan 10** if you order in the next **8 hours** and choose one-day shipping.

(3,589)

Eligible for **FREE** Super Saver Shipping.

Books: See all 31,343 items

HIIS LAB ISTI CNR. DESIGN BY CSS TEMPLATES

Fig. 4. The resulting mashup

4 The Architecture of the Environment

Our mashup environment is based on a server with twofold functionalities: it acts as a proxy and provides support for the authoring environment, including session management functionality for managing multiple users at the same time. Thus, the access to the Web applications should go through the proxy/mashup server, which parses the original document and adds the ids to those relevant elements that do not have one. The relevant elements are the ones that can be interactively selected (such as DIV, FORM, INPUT, TABLE, …). The server also includes a number of scripts, which are exploited by the mashup editor, to all the Web pages that are accessed through it. Such scripts allow the users to interactively select the elements to include in the newly created mashup. If the element is a container (e.g. a div) its inner components are recursively included into the selection. The elements selected in the Web page are highlighted by changing the background colour. To this end, the scripts that are automatically included are able to manage the on mouse over and on click events. If there are already some handlers for such events in the Web application, they are preserved and such behaviour is added to them. In particular, hovered components are gray-highlighted, selected components are green-highlighted. Highlighting of a component is done by changing its background and border colour. The original attributes value is saved on a JavaScript variable in order to be restored when the component is unselected or unhovered.

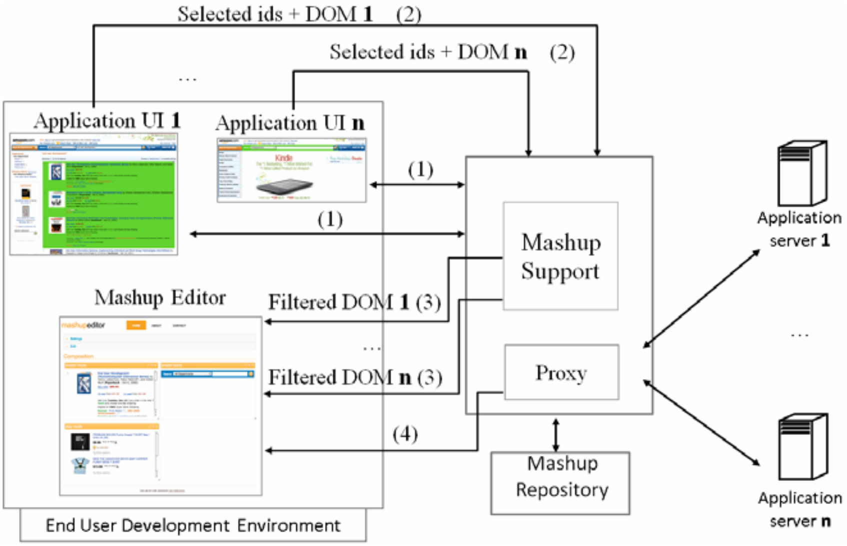


Fig. 5. The Underlying Architecture of the Mashup Environment

When the users are satisfied of the elements selected in a given application, they can send them to the mashup editor. In this case when the transmission is triggered the injected scripts send the ids of the selected elements along with the DOM description of the considered Web page to the mashup server (2 in figure 5). The mashup editor assumes that each DOM node has a unique id, and the proxy server supports this by first adding the missing ids to the original HTML page. The DOM sent to the mashup editor is state-persistent: this means that, for instance, it maintains the values previously entered by the user in a form. In the server the DOM is filtered removing all the elements that are not selected. The resulting filtered DOM is sent to the mashup editor (3 in figure 5), which allows the user to interactively determine the position and the dimensions of the set of new elements in the mashup layout. It also asks the user for a title to the newly added component. The list of the selected elements identified is also kept in the mashup server in order to allow the update of the components, as explained later on.

This type of selection of Web components and their inclusion in the mashup can be performed on various Web applications going through the same process.

The grid indicating the set of components in the mashup editor is stored in the associated session. In the stored information it is indicated where each component is located in the mashup layout; the path to the HTML file created in the mashup server, which is included in an internal frame; the list of parameters that the user has selected as possible values that can be provided by other components; the list of input parameters that the user can enter and that can be sent to other components.

In order to create communication among the groups of elements selected, the environment allows the identification of their communication needs. They are mainly classified into elements that require input from the user and can produce an output that can be sent to a Web server (input components), and elements that can receive input from the Web server and exploit such input to generate output towards the user (output components). It is also possible that one component has both input and output capabilities, but we will maintain the role distinction for the sake of clarity. The mashup environment is able to create connections between these two types of elements even if they originally belong to different applications. For this purpose, on user request, it is able to show the list of parameters that determine the content of output components showing query results and asks the users to select the elements that are the input that they entered to obtain them (the other parameters are generated by the application for its state management). Then, the user can activate a connection manager, which is able to allow the user to indicate what input components should be used to provide the parameters that determine the content of the output components.

The mashup environment is able to intercept the HTTP requests that in the case of a GET method use query strings and in the case of a POST method use post payloads. In both cases, the parameters are encoded according to a standard syntax (URL encoding), which enables its automatic analysis. Thus, by identifying all the parameters associated with an output component it is then possible to allow users to select any of them and indicate that the value entered in a given field in an input component should feed the selected input parameter. In order to perform such connection, the environment is able, on user demand, to show a list of the input fields that are contained into a mashup component (e.g. text fields, drop down lists, text areas etc.). For each field, the environment shows the list of input parameters for output components that can receive the value entered by the user through it. Figure 3 shows an example of such connection procedure: the input component contains two fields, a drop down list and a text field. The drop down list is not connected to any parameter, while the value entered in the text field is connected to the *amazon_keyword* of the *Amazon Result* component and to the *ebay_keyword* parameter of the *Ebay Result* component. This means that when the form containing the text field is submitted, the mashup editor updates the *Amazon Result* and the *Ebay Result* component, changing the value of the parameters in the query string associated to *amazon_keyword* and *ebay_keyword* respectively. For this purpose, through an analysis of the DOM, the action attributes of all forms are modified by the mashup server in order to submit to the mashup environment. The update procedure follows the following steps:

1. The mashup environment collects the value inserted by the user in the form;
2. For each value collected the components that should receive it are marked for update;
3. For each component that needs an update the new URL or POST content is calculated with the parameter value changed;
4. For each component that needs update, the mashup environment downloads in the server the full updated web page, filters the DOM using the id list created when the user selected the elements for creating the component, and uploads the result in the mashup editor.

The environment also allows the users to save the result of their mashup editing for further reuse by themselves or other users. This information is stored using the MARIA [9] logical language for interactive applications. The parameters are stored in the data model of the resulting MARIA specification. The layout is specified using a grid grouping composition operator of the language by defining the corresponding internal frames sources, width and height attributes.

One further advantage of saving the mashup specification in MARIA is that it is then possible to adapt it for access through mobile devices. For this purpose, it exists a tool for desktop-to-mobile adaptation [10], which performs an adaptation taking into account the screen size of the target device and other aspects.

5 Evaluation

A usability test has been carried out in order to evaluate the prototype of our mashup environment and, in general, to collect feedback from users.

12 subjects were involved (2 females, 10 males), with age ranging between 27 and 41 year old (average 30.3). The users were all recruited within the personnel of our research institute. However, among them there were not only researchers but also technical/administrative collaborators and university students. Furthermore, the participants were characterized by diverse educational level: 2 had PhD, 7 Master Degree and 3 Bachelor Degree.

With respect to the knowledge of programming languages, in a 1 to 5 scale (with 1 as most negative and 5 as most positive score), it varied between 2 and 5, (average 3.75, standard deviation 0.9).

It is worth noting that, as already mentioned, not all the users were equally experienced with programming. Four of them declared not to have any experience with the most common web-oriented languages (HTML / JavaScript). Furthermore, two users had only very little knowledge of C/C++ and some experience with Matlab. Only two users declared they had previously used a tool for creating web mashups (iGoogle).

The participants were required to interact with the mashup environment, through a desktop PC, for creating the mashup application introduced beforehand. In detail, they had to perform two searches: one in Amazon and one in eBay, respectively. Then, they had to select and move towards the mashup editor the Amazon search bar, a couple of Amazon results and one of the eBay results. Layout customization was then possible, and the users could resize and relocate the three widgets according to their preferences. At that point, the mashup consisted only in a presentation composition of Web components. One important part of the test was the binding between the input component (text field of Amazon search form) and the output components (eBay results and Amazon results). The binding is necessary to fully exploit the mashup environment by enabling the possibility to compose the functionalities of the components selected from the various Web applications.

Another relevant aspect of the test was to let the users notice the relationship between the originally selected component(s) and the ones resulting from the interaction with the mashup. At the end of the interaction, the participants were indeed requested to accurately observe the content of the widgets and to ensure about their correspondence with the parts extracted from the original pages. In detail, they had to look at the position of the search result items of Amazon and eBay with respect to the original page (this was possible by looking at the item index). Thus, besides the simplicity of the example, the users could realize the possibility of selecting additional and diverse parts of the original pages, such as the footer bar of Amazon or the left menu of eBay, to bring them to the mashup and to keep them while interacting with it.

The interaction took between 10 and 15 minutes per user, during which the number of errors was recorded together with their type, distinguishing between interaction errors and conceptual errors: general ones are related to errors in the interaction with the user interface (e.g., wrong selection in the source web page due to difficulties in moving the mouse pointer on the right element); conceptual ones are due to misunderstanding in the application of the mashup editing procedures (e.g., selecting only one variable in the binding table).

Nine users made at least one error. With respect to interaction mistakes, 5 users made 1 error each. With respect to conceptual mistakes, 3 users made 1 error each and other 3 users made 2 errors each.

Afterwards, the users filled in a questionnaire for rating several aspects of the mashup environment in a 1 to 5 scale (with 1 as most negative and 5 as most positive score). In the following, the rating for the intuitiveness/simplicity of the various tasks performed is reported ([min; max], average, standard deviation):

- Opening a new navigation window ([2; 5], 4.1, 0.9).
- Selecting the components on the navigated page and sending them towards the mashup tool ([3; 5], 4.3, 0.7).
- Instantiating and labeling the new widget ([3; 5], 4.7, 0.6).
- Rearranging (moving, resizing) the instantiated widgets in the mashup environment ([2; 5], 4.3, 0.9).

- Finding, in the connection panel of the mashup environment, the data previously filled in the form of the search page ([3; 5], 4.0, 0.8).
- Mapping the parameters of the two original applications in the connection panel ([2; 5], 4.2, 0.9).

Some further answers or indications were also provided in the questionnaire regarding particular aspects of the implemented prototype or with respect to recommended modifications/improvements. As weak point, regarding the interface for selecting and renaming the previously inserted parameters, one user stated that listing too many parameters could be confusing. He then suggested to automatically restrict, in some way, the parameters to be indicated in the connection table.

Another user declared that, using just the strings for identifying the parameters, can be difficult sometimes. Thus, she proposed the usage of visual techniques for recalling them.

Others suggested to further improve the support for the user, such as automatically pop-up the mashup editor window after one component is sent from a Web application to the mashup editor.

Even those users who initially had not well understood the actual meaning of the mashup editor, at the end of the interaction seemed to realize its capabilities and usefulness.

We investigated also the potential relationships between the programming experience of the users and the number/type of interaction mistakes. However, such relationships did not emerge from the test results. Indeed, we observed that general and conceptual mistakes occurred both among experts and non-experts users. At the same way we noticed that, among who did not make any error, there were users with high as well as with low programming experience. Relationships between experience and mistakes do not emerge even distinguishing the kind of user programming experience (i.e.: whether the user had familiarity with specific web-oriented programming languages). Thus, due to this apparent lack of relationships, we are prone to assert that the user attitude towards our mashup environment does not depend on the user programming experience.

6 Conclusions and Future Work

Despite the enormous number of Web applications often users need to create new composite applications that better suit their needs. In this paper we have presented the motivations and the design of a new solution that aims to allow even people who are not professional programmers to create their Web mashups. We have described an example application, the underlying architecture and the implementation of a prototype system, which has already been tested by a set of users. We also report on this first user test, which has provided some useful suggestions, for example to provide some graphical metaphor to make more intuitive the connection mechanism among components.

The interesting point of our approach is that it allows users to create the mashups through direct manipulation of existing Web applications without requiring the use of a specific browser and the knowledge of any programming language. This is obtained by automatically annotating the Web pages that are included in the environment by some scripts when they pass through our server. We are considering changing the technique used for identification of the various Web page elements that can be selected by the users in order not to relay on DOM nodes ids. A solution based on the XPath node should be more general.

We also plan to perform more systematic user validation of the proposed environment by involving a wider set of user groups, possibly without any development experience.

References

1. Baudisch, P., Xie, X., Wang, C., Ma, W.: Collapse-to-zoom: viewing web pages on small screen devices by interactively removing irrelevant content. In: UIST 2004, pp. 91–94 (2004)
2. Dannecker, L., Feldmann, M., Nestler, T., Hübsch, G., Jugel, U., Muthmann, K.: Rapid Development of Composite Applications Using Annotated Web Services. In: ICWE Workshops 2010, pp. 1–12 (2010)
3. Hartmann, B., Wu, L., Collins, K., Klemmer, S.R.: Programming by a sample: rapidly creating web applications with d.mix. In: UIST 2007, pp. 241–250 (2007)
4. Li, P., Wohlstadter, E.: Script insight: Using models to explore JavaScript code from the browser view. In: Gaedke, M., Grossniklaus, M., Díaz, O. (eds.) ICWE 2009. LNCS, vol. 5648, pp. 260–274. Springer, Heidelberg (2009)
5. Lin, J., Wong, J., Nichols, J., Cypher, A., Lau, T.A.: End-user programming of Mashups with Vegemite. In: IUI 2009, pp. 97–106 (2009)
6. Lieberman, H., Paternò, F., Wulf, V. (eds.): End User Development (2006) ISBN: 978-1-4020-4220-1
7. Maras, J., Štula, M., Carlson, J.: Extracting Client-Side Web User Interface Controls. In: Benatallah, B., Casati, F., Kappel, G., Rossi, G. (eds.) ICWE 2010. LNCS, vol. 6189, pp. 502–505. Springer, Heidelberg (2010)
8. Oney, S., Myers, B.: Firecrystal: Understanding interactive behaviors in dynamic web pages. In: VLHCC 2009: Proceedings of the 2009 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), pp. 105–108. IEEE Computer Society, Los Alamitos (2009)
9. Paternò, F., Santoro, C., Spano, L.D.: MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. ACM Trans. Comput.-Hum. Interact. 16(4) (2009)
10. Paternò, F., Zichittella, G.: Desktop-to-Mobile Web Adaptation through Customizable Two-Dimensional Semantic Redesign. In: HCSE 2010, pp. 79–94 (2010)
11. Yu, J., Benatallah, B., Casati, F., Daniel, F.: Understanding Mashup Development. IEEE Internet Computing 12(5), 44–52 (2008)
12. Zang, N., Rosson, M.B., Nasser, V.: Mashups: who? what? why? In: CHI Extended Abstracts 2008, pp. 3171–3176 (2008)
13. Wong, J., Hong, J.I.: Making mashups with Marmite: towards end-user programming for the Web. In: CHI 2007, pp. 1435–1444 (2007)

Alternative Representations for End User Composition of Service-Based Systems

Usman Wajid, Abdallah Namoun, and Nikolay Mehandjiev

Manchester Business School, The University of Manchester
{usman.wajid,abdallah.namoune,nikolay.mehandjiev}@mbs.ac.uk

Abstract. Service composition is a process of developing service-based applications that combine the functionality and features from multiple service providers in a unified solution. In this paper we report on a study aimed to gauge users' views and perceptions about traditional service composition approaches (such as control flow and data flow-based composition approaches) versus a system assisted composition approach. User preferences and opinions are obtained from a set of focus groups that aimed at exploring the mental model of end-users about the way they would prefer to develop service-based applications. The results of user studies are being used in the design of an easy to use service-based application development tool in the EC funded SOA4All project.

Keywords: focus groups, service-based systems, service composition, control flow, data flow, system driven, end-user development.

1 Introduction

Service-based systems are useful for developing applications that combine the functionality and features from multiple service providers. In these systems individual services are used as building blocks bringing the expertise and core competencies from various entities into one unified solution to address a specific purpose/requirement. Key benefits of application development in service-based systems include producing combined functionality which is not yet available as a service, improved reusability because the same service can be part of many applications and improved flexibility since modifications can be made by replacing an individual service in a service-based application. However, despite the advantages of service-based systems only a small proportion of users, often with considerable modelling and programming skills, can develop service-based applications. The majority of (Internet) users are unable to exploit the benefits of service-based technologies and create service-based applications tailored to their specific needs. This limitation can be attributed to the complexity of the existing application development approaches and the limited technical knowledge of ordinary end users. Thus, the research challenge lays in simplifying application development and abstracting this process from any unnecessary technical complexity, with the general aim of promoting service reuse and consumption especially among end users who have very limited or no technical background. In this

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 53–66, 2011.

respect, the EC-funded project SOA4All[1] addresses this research challenge by developing a framework and a set of tools to support the service lifecycle from service discovery to composition and consumption. The centrepiece of SOA4All tools is SOA4All Studio, a web platform that will provide users with a unified view covering the whole lifecycle of services, including design-time, run-time and post-consumption analysis.

In order to engage end users in the process of service-based application development this paper highlights the significant details of three application development approaches, namely control flow, data flow, and system driven application development approaches. We then report on the user opinion about these approaches.

A comparison between manual (control flow, data flow) and system driven approaches is likely bring up unsurprising results. However, in our daily lives we see many examples where users prefer manual approaches over automated or system driven alternatives. For example, it is common that users choose to sort their email manually in MS Outlook, although Outlook provides automated filtering mechanism for sorting emails. Moreover, web designers often choose to make modifications in their Web pages in *code view* rather than using *design view* of webpage editors e.g. MS FrontPage.

In this respect, for us user opinion and preference is important in making an informed selection on a (user preferred) approach that can be used in the service-based application development tool, which will become an integrated part of SOA4All studio. In essence, the selection of a user preferred approach will allow end users to develop composite service-based applications tailored to their needs and requirements.

The remainder of this paper is organized as follows: Section 2 reviews the existing work on service-based application development by end users. Section 3 presents the procedure of focus groups and evaluation methodology. Section 4 reports the perspective of end users on service-based application development using the three alternative approaches and Section 5 concludes the paper.

2 Development of Service-Based Systems

At present Internet users can add independent web services as widgets to their personalized pages on iGoogle[2] and Facebook. However, this cannot be regarded as application development because users cannot wire services together to exchange and share data or functionality. The produced (service-based) personalised pages are very trivial and offer autonomous services. It is more useful and interesting if end users are not only enabled to produce rich and complex service-based applications that fulfil their specific needs, but also allowed to easily extend and customise such applications. An integrated mini holiday-booking application which consists of a flight-booking service, a hotel-booking service and a car-booking service is a good and realistic example of an interactive service-based application that users can build and customise by adding or removing specific services.

[1] http://www.soa4all.eu

[2] http://www.google.com/ig

A variety of mashup builders, ranging from data flow, UI/widget, and event-based, exist to enable end users with no programming skills to create mashups or software applications that integrate data, logic, or user interfaces in order to meet user's evolving needs [e.g., 17, 20, 22, 1]. However, these tools are not as easy to use as they should be since they have high learning curves. In this respect, end users are entailed to spend a lot of learning time and efforts in getting used to, including reading tutorials, practising demos and examples, and sometimes understanding even programming and modelling concepts such as loops [18]. Among existing tools, an interesting mashup builder is Yahoo Pipes![3], which empowers users to combine various information sources and perform different filtering operations to achieve a desired outcome. The major drawback of Yahoo Pipes! is its reliance on user modelling skills which most end users do not have. In addition, the nature of information available to users and the type of operations that users can perform on that information is quite limited. This motivates further work to deliver non-programmers easy to use and simple tools that are sufficiently powerful to create rich service-based applications.

Existing research efforts in this area mostly focus on the development and implementation of latest technologies and modelling or programming languages to facilitate and realise application development. However, such initiatives are typically targeted towards business and software developers, with virtually no attentions to the ordinary end-users' needs and perspective. Moreover, existing approaches for service-based application development typically employ top-down techniques where the users are expected to get familiarized and use the approach that has been presented to them. For example, the surveys in [1] and [2] discuss existing approaches with a particular focus on the technical aspects such as the use of various languages (e.g. OWL-S[4], BPML[5]) and technologies (e.g. Petri-nets) employed by the existing approaches. Nonetheless we believe if ordinary users are to capitalise on the benefits offered by Service Oriented Architectures (SOAs), the focus should not be on the technical aspects, but on how these are presented and how users can use these advanced technologies to perform their desired tasks at a minimal cost.

The discipline of *End User Development* (EUD) [6, 7, 8] has long-standing traditions in allowing users who are not programmers to design and modify non-service-based software. Some practical successes include spreadsheets, macro recording and database form painters, and some theoretical advances in the cognitive design of end user development environments, and studies of representations and problem-solving models of end users acting as developers. It is now clear that such end user development environments should be tuned to the target domain and tasks of the end users, and they should employ representations which are oriented to the skills and background of end users. For example, *spreadsheets* are hugely successful in the domain of mathematical computations because they employ the textual language of formulae and the metaphor of accounting tables to organise computations.

To ensure this is the case within our tools (within SOA4All) i.e. tools for application development in service-based systems; we need to start with understanding of user preferences and issues regarding different representation styles. This approach is

[3] http://pipes.yahoo.com/pipes/

[4] http://www.w3.org/Submission/OWL-S/

[5] http://www.ebpml.org/bpml_1_0_june_02.htm

consistent with the studies we have undertaken previously [8, 9] in establishing the balance between costs and benefits for end users when they undertake *end user development* – developing or modifying software. Identifying end users' expectations in regard to the perceived risks and benefits of using new software applications is a good indicator of potential uptake of end user development activities [9]. Namoun et al [16] conducted a series of focus groups in which end users (ordinary web users and IT professionals) discussed the risks and benefits of service composition. Users' greatest worries were centred around security and privacy issues regarding their personal information. However, their interest and motivation to compose services were quite high and promising.

The studies underpinning this approach are generally known as studies of *"natural programming"* practices [10]. Such studies have been applied to many areas ranging from end user development of simple game applications by kids to studies of debugging practices of professional programmers. The applications to the domain of services, however, are only a few. For instance, Beaton *et al* [11] study programmers when they attempt to resolve problems linked to a library of enterprise service APIs, and demonstrate the overly complex nature of these APIs and the lack of documentation making progress difficult. However, to the best of our knowledge, there are no other reports of studies of the notations end user developers use when attempting to develop service-based applications.

Although mashup and composition research areas have advanced a lot in comparison to the old hacking practices, only a limited number of studies focused on understanding the needs of ordinary end users and improving the usability of development environments. Cao et al [21] applied a design theory-based approach to end user programming activities to analyse and understand the problems end users run into while creating mashups and their problem-solving strategies. Namoun et al [19] also performed a set of user testing, highlighted the main problems and devised a number of corresponding guidelines for lightweight service composition, such as support for task-based composition, graphical direct manipulation of components, and high level of abstraction. Similarly but mainly depending on the actual users' conversations posted in Yahoo! Pipes forums, Jones and Churchill [1] analysed the discussion threads and extracted the main problems Yahoo! Pipes users face when they develop web mashups, primarily the inability to localise faults.

The majority of mashup and composition tools are based on little, if no, scientific foundation or evidence for the concepts and features used within them. A more plausible design strategy is to consider diverse design alternatives for composing services or mashing up data and involve actual end users as well as designers into analysing these alternatives in order to make well-informed design decisions.

Despite rapid developments in service-oriented technologies, service-based application development by end-users is an area in its infancy. In this respect, identifying the needs and specific requirements of ordinary users is a crucial prerequisite to the design of an *"easy to use"* and *"easy to understand*" application development approach. Such a bottom-up approach can be helpful in promoting the uptake of service-based technologies by ordinary users.

3 Focus Groups to Evaluate Three Design Alternatives for End User-Based Composition of Service-based Systems

3.1 Method

In our study we apply design rationale in order to understand and guide the impacts of differing composition approaches on end user development. Design rationale is an HCI technique used to document and analyse various design alternatives to a certain interaction problem and carefully investigate the reasons/arguments behind design decisions [15]. In design rationale, features of the system-in-use with positive or negative consequences are often enumerated and analysed. Such kind of analysis enables designing new effective systems containing features to mitigate the downsides and enhance the upsides of the system-in-use.

We performed a set of collaborative workshops with representative end users where we showed them the potential design alternatives using Microsoft PowerPoint, provided an in-depth explanation of each development approach and how it works, and invited them to explore and discuss the various advantages and disadvantages of each design and reason about the tradeoffs associated with the design features. We adopted this approach for three main reasons, namely to:

1. Analyse the design features of each particular design and directly link their consequences to user's interaction behaviour in order to define the argumentations that led to the design decision
2. Improve the upsides and eliminate the downsides of the design alternatives
3. Evaluate the trade–offs and define other alternative designs

Users' discussions, comments, and opinions about the three design alternatives were recorded using an audio recorder; these discussions were then transcribed for follow up analysis. Transcripts were analysed by two human factors experts using thematic analysis [14], an inductive method for analysing qualitative data by defining the appropriate themes or patterns which emerge from the transcribed data; these themes are not imposed by the analyser. The emerging themes as regards service development approaches are highlighted and explained in the results' section.

3.2 Procedure

For the purpose of exploring and comparing three design choices to end user-based application development we organised three separate focus groups which included, in total, 35 end users with no IT-background or modelling skills. The focus groups were held in different times of the year and each of which contained the same procedure. Focus group is a qualitative technique used to generate data and opinions on a certain subject from potential users of a particular system [3]. In details, all participants were instructed to complete the following tasks:

1. Fill in a demographic questionnaire about their age, IT level, software and service composition experience
2. Listen to a 20 minute presentation which introduces application development by means of service composition alongside some examples (e.g. a google map *mashup* showing news headlines at relevant places on the map of Europe)

3. Discuss the feasibility of composing various services in an application, its associated risks and benefits. Here, it is worthwhile to note that the moderators of the focus groups stayed neutral during the discussions and did not express their opinions in anyway
4. Walk through each design alternative using a low fidelity prototype and identify the merits and problems of each design in small groups of 5 participants
5. Rate the three design alternatives by completing a subjective rating questionnaire

3.3 Materials

The user opinions about three alternative application development approaches have been gathered to simplify the application development process and to hide the underlying technical complexity from ordinary users. These approaches are inspired by (1) engineering methods for analysing computer programs and (2) human interface design methods. To demonstrate and explain the features of the three approaches we have developed three low-fidelity prototypes which we presented to our target end users using Microsoft Power Point. An overview of each of the three approaches alongside a figure is given below.

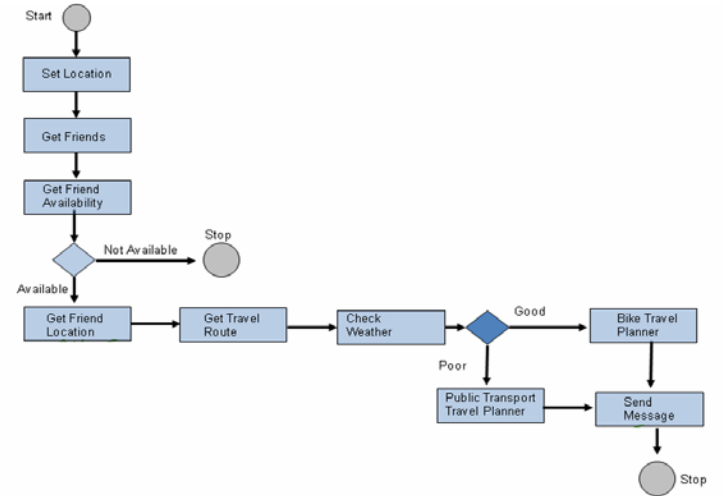


Fig. 1. Control flow-based application development approach

3.3.1 Design One: Control Flow-Based Application Development Approach

Control flow (an example shown in) represents a sequential application development process where a task or service requires completion before the next task / service is executed. In other words, control flow defines the order in which atomic steps or services are executed. Moreover, control flow approach enables users to define various useful relations between services, namely: (1) unconditional branching, (2) conditional branching, (3) iterative execution (i.e. loops), and (4) unconditional stopping.

Unconditional branching is simply a continuation to the next service. Conditional branching is specified using predicates, for instance: GetFriendLocation service (in), which retrieves the exact location of friends, is executed if and only if, GetFriendAvailability service satisfies a precondition "Friends are available". Iterative execution defines the execution of a particular service zero or more times until a condition is satisfied. Unconditional stopping signifies the end of the composite application.

This approach does not deal with how and what type of information will be passed between single services.

3.4 Design Two: Data Flow-Based Application Development Approach

Data flow (an example shown in) represents an information-oriented view of application development, where data is passed between multiple services without the requirement of a specific sequence. Data flow diagrams typically contain a set of concurrently executing services that exchange messages. This approach enables users to define how data flows from source service(s) to destination service(s). Each connection in the diagram ought to carry the data that is passed from one service to another. Using this approach, no information about service execution order and conditions can be defined or elicited

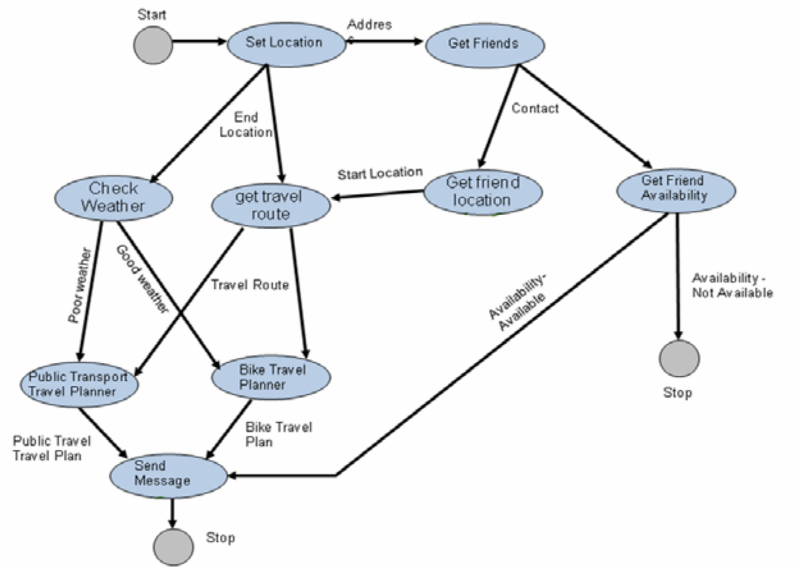


Fig. 2. Data Flow-based application development approach

For example, in the data from Get_Friend service is passed on to Get_Friend_Location and Get_Friend_Availability services without any preference or condition for execution order of later services.

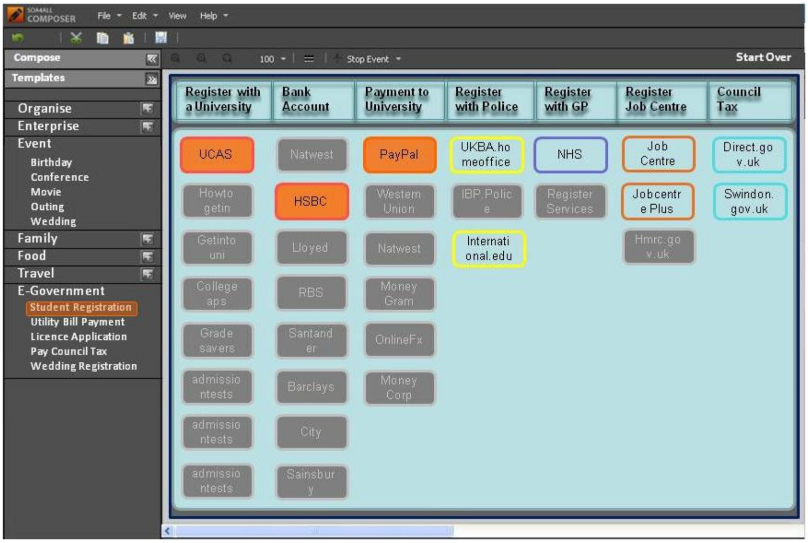


Fig. 3. System driven application development approach

3.5 Design Three: System Driven Application Development Approach

The system driven approach is drawn from human-computer interface design methods and principles. This approach uses domain specific templates, which are organised in various categories. In this respect, this approach enables users to choose individual services from a wide range of available application templates organised in various categories.

Once a user selects a template, the underlying reasoning mechanism selects appropriate service instances matching their functionality, inputs and outputs [12], and positions them under the appropriate activity, or task, from the template. As shown in an application for 'International Student Registration' include the tasks of 'Register with a University' and 'Bank Account', etc. The user can then select one service per task, and the underlying reasoning mechanism will grey out all services which are incompatible with the user selection in all other tasks of that application, thus ensuring that only services with compatible inputs and outputs are selected.

For example if a user selects a service UCAS' from 'Register with a University' task then only compatible services are highlighted in the subsequent tasks and incompatible services are greyed-out. Users can select from a wide range of customisable system templates and do not have to define control and data flow among services as these aspects are managed automatically. In this respect, shows user selections for the first three tasks, and greyed out incompatible services for the remaining tasks.

4 Results

The results reported here are extracted from three focus groups organized with non-programmers (35 participants in total) to capture their perceptions about the three application development approaches. The participants discussed the advantages and disadvantages of each approach from an end user perspective, followed by filling in rating questionnaires. We used inductive content analysis [2] to analyse the qualitative results gathered in the focus groups.

Our participants (age mean 26, including 13 males and females 22) had a relatively weak software development experience (2.31 (std= 1.13), ratings were performed on a 5-point rating scale where 1=none and 5= expert) and service development experience (2.11 (std=1.15)). Most users reported experience using Facebook, iGoogle, My Yahoo, and Hi5. These systems were thought of as examples of service-based application development platforms and languages. Only three users reported the use of Yahoo! Pipes and two other users reported the use of OWL-S and BPML. These results demonstrate that our users were suitable representatives of ordinary web users.

Regarding the general notion of "end user-based application development in service-based systems" participants appreciated and received well the idea of developing their own applications. They were motivated by several reasons mainly owing to the prospect of being able to tailor service based applications based on their diverse needs, no need for learning or writing programming code, saving time and avoiding errors resulting from manual calculations, convenience and ease of use. However, despite their optimism end users were worried about the security and privacy of their personal information especially in applications that require sharing or publically declaring the sensitive information such as bank details. They were also concerned about using and trusting services provided by third parties or services developed by unknown (or not well know) developers/companies. Other concerns centred on the complexity of the application development process and the efforts required achieving it.

4.1 Comparison of Three Application Development Approaches

The overall results show that users liked *control flow*-based approach for its ease of use. The simplicity of the approach was admired by the participants of our study. However, users expressed concerns about the strict nature of the approach as it does not allow them to drag and drop services in an application without revising the overall application. Also the participants were doubtful about how control will be passed through one service to another, thus the approach was conceived as simple but requiring attention to details to make it work. Additionally, participants argued that due to the sequential nature of a control flow an application involving various services can get too complex for them to keep track of.

On the other hand, *data flow*-based approach ranked high for the flexibility it offered to users, but was regarded as a difficult approach for non-programmers due to the underlying complexity of managing different data sources. Moreover, when a data flow diagram (in Figure 2) was shown, some participants pointed out that it was difficult for them to understand how the overall structure works. The complexity of the data flow diagram added to the negative scoring for this approach. Other opinions reflected similar worries about the disorganized representation of data flow and the resulting complexity in understanding relationships between various services. Hence, data flow approach was regarded as more suitable approach for users with technical background and programming experience.

In the end, the *system driven* approach came out as the easiest approach to use and scored the highest ranking for acceptability by the participants. Overall participants were highly interested in the system driven approach, since it reduces the complexity of application development by providing a systematic way of binding compatible services in an application. The interest in system driven approach can be linked to the non-programming background of users, as most of the users regarded control flow and data flow as ”programmer’s way of building software artefacts”. In this respect, participants preferred system driven approach over the other two approaches because it saves time and hides the complex technical details from the user.

However, despite being the preferred choice of majority of participants in our studies, few issues were raised during the discussions about the scalability and flexibility of the system driven approach. For instance, participants showed interest in customising and creating their own templates and extending the list of available services that appear by default in each template. In addition, questions were raised about the scalability of the approach in accommodating many services on a template and how the approach will be able to accommodate user preferred services in a template that are not available in the list of services.

Moreover, probing the limited flexibility offered by the approach participants were interested if the approach could allow them to select more than one service for a single task, For example, rather than getting contact details from one service provider i.e. Gmail, the user were interested if they could add some contacts from other service providers as well such as Facebook.

In the remainder of this section we have analysed the results of focus groups along with the discussion that took place during the user studies. The discussions were of interest to us as they revealed the user opinion about each approach in an informal manner, thus allowing participants to freely discuss the merits and limitations of each approach with others.

Here (in Table 1) we provide a breakdown of user opinion as merits and problems for each application development approach.

Table 1. Users’ feedback about the three application development approaches

Design approach	**Merits (+)**	**Problems (-)**
Control flow	• Enables users to follow the logic and execution steps of the process or application in a linear representation • Easier to detect problems using this type of diagrams	• Designed for programmers • Compatibility issues between various components or services • Depends on trial and error • Limits user choice • Requires good analytical skills

Table 1. *(continued)*

Data flow	• Shows the data that flows between services (i.e. shows what it is required –input- and what is the result -output-) • Gives in-depth knowledge about the process	• Complex representation • Designed and well suited for programmers or people with technical background • Limits user choice • Confusing • Depends on the modelling skills of the users • Difficult to establish data flow with new services
System driven	• Services can be manipulated • Shows only the compatible services based on user selection • Easy to use, easy to modify, understand and follow • Organises information and services within a template • User-friendly interface • Suits the profile of ordinary end users and allows them to develop applications in a simple manner • Provides templates to guide the user in accomplishing their goals • Services are presented according to the goal they fulfil	• Interoperability and compatibility: this approach has to deal with data transfer and compatibility issues between various services • Privacy and security of data: authorisation and validation of data to be used by the user is one major concern • Limited user control and less options for the user to choose from. Therefore this approach is not suitable for programmers. • Users need to customise the templates to fulfil their various needs

Table 2. Users' subjective rating of the three application development approaches on a 5-point rating scale

Usability Measures	**Control flow**	**Data flow**	**System Driven**
Easy to use	4 (std 1.09)	3.45 (std 1.12)	4.27 (std 0.94)
Easy to understand	4.81(std 0.40)	3.45 (std 1.21)	4.09 (std 1.00)
Effective	3.63 (std 1.28)	3.36 (std 1.02)	4.27 (std 0.78)
Overall rating	4 (std 0.89)	3.54 (std 0.82)	4.45 (std 0.82)

Subjective ratings showed that system driven ranked higher than control flow and data flow-based approaches on all usability measures, except "easy to understand", as shown in Table 2. Moreover, the data flow approach was perceived as the most complex approach and received the lowest ratings. This result emphasizes the need to simplify existing application development approaches which mainly rely on data aggregation such as Yahoo! Pipes. It is worthwhile to note that all questions were rated on a 5 point rating scale where 1=disagree and 5=agree apart from the last question (overall rating of the current approach) where 1= bad and 5=good.

5 Conclusions and Lessons

The paper reports on the user opinions about three alternative application development approaches namely control flow, data flow and system driven approach. The aim of the overall user study was to identify approaches that can enable ordinary users, who have no significant modelling and programming skills, to develop service-based applications. Focus groups revealed control flow was ranked as the easiest approach for end-users to understand but using the control flow approach was identified as problematic for developing applications of realistic complexity, because of the need for keeping track of order of activities and also the need to understand and manage the hidden interactions between services in terms of data.

The discussions during the focus groups pointed that end users favour system-driven approach owing to its ease of use. Whilst both control flow and data flow approaches require profound knowledge of programming and modelling concepts, system driven approach hides away all the technical details and complexities because of the automation we envision in matching compatible services. This vision is not without a solid theoretical foundation; indeed work on automatic calculation of input and output compatibilities between services has already been reported [13], this can easily provide the necessary fit between service instances and their placeholder activities in a service-based application template [12]. Nevertheless, for the later approach to succeed it has to offer the right level of control to users by allowing them to customise templates and add services according to their needs.

Other recommendations from our user studies include empowering users to rate, share comments, and recommend desired services. Users should also be allowed to modify and set security levels for their applications and should be helped through constant system support and notifications in case of problems. Moreover, some users suggested the combination of both control flow and system driven approaches, which can allow users to edit the sequence of services whilst still being guided by the system, such as approach can provide support for developing more powerful composite service-based applications.

We believe that these requirements are important for designing tools that realise the benefits of end-user based application development in service-based systems. In this respect, our study has formed an important first step towards creating a robust and user-friendly environment allowing effective development of service-based applications by people with little or no experience in software development. This addresses the dearth of such studies in the literature, and has provided a valuable input to the development of service-based application development tools in SOA4All Studio.

References

1. Beek, M.H., et al.: A Survey on Service Composition Approaches: From Industrial Standards to Formal Methods. Technical Report. Istituto do Scienza e Tecnologie dell'Informazione. 2006-TR-15 (2006)
2. Dustdar, S., Schreiner, W.: A Survey on Web Service Composition. IJWGS 1(1) (2005)
3. Hsieh, H.F.: Three Approaches to Quantitative Content Analysis. Qualitative Health Research, 1277–1288 (2005)
4. Marshall, C., Rossman, G.B.: Designing Qualitative Research, 3rd edn., p. 115. Sage Publications, London (1999)
5. Lieberman, H., Paterno, F., Wulf, V. (eds.): End User Development. Human-Computer Interaction Series, vol. 9, p. XVI, 492. Springer, Heidelberg (2006)
6. Mehandjiev, N., Sutcliffe, A.G.: Guest Editors. Journal of Organizational and End User Computing 18(4), 43–65 (2006); Special Issue on Technology Interaction Aspects of End User Development (2006) ISSN: 1546-2234
7. Sutcliffe, A.G., Mehandjiev, N.: Guest Editors. Communications of ACM, a Special Issue on End User Development (September 2004)
8. Mehandjiev, N., Sutcliffe, A., Lee, D.: Organisational View Of End-User Development. In: Lieberman, H., Paterno, F., Wulf, V. (eds.) End User Development. Human-Computer Interaction Series, vol. 9, p. XVI (2006)
9. Mehandjiev, N., Stoitsev, T., Grebner, O., Scheidl, S., Riss, U.: End User Development for Task Management: Survey of Attitudes and Practices. In: Proceedings of 2008 IEEE Symposium on Visual Languages and Human-Centric Computing, Herrsching am Ammersee, Germany, Herrsching am Ammersee, Germany, September 16-20. IEEE Press, Los Alamitos (2008) ISBN : 978-1-4244-2528-0
10. Myers, B.A., Pane, J.F., Ko, A.: Natural programming languages and environments. Communications of ACM 47(9), 47–52 (2004), http://doi.acm.org/10.1145/1015864.1015888
11. Beaton, J., Sae, Y.J., Yingyu, X., Stylos, J., Myers, B.A.: Usability challenges for enterprise service-oriented architecture APIs. In: IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2008, pp. 193–196 (2008)
12. Mehandjiev, N., Lécué, F., Wajid, U.: Provider-Composer Negotiations for Semantic Robustness in Service Compositions. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 205–220. Springer, Heidelberg (2009)
13. Lécué, F.: Optimizing QoS-Aware Semantic Web Service Composition. In: International Semantic Web Conference 2009, pp. 375–391 (2009)

14. Braun, V., Clarke, V.: Using thematic analysis in psychology. Qualitative Research in Psychology 3(2), 77–101 (2006)
15. Moran, P.T., Carroll, J.M.: Design Rationale: Concepts, Techniques, and Use. Lawrence Erlbaum Associates, New Jersey (1996)
16. Namoun, A., Wajid, U., Mehandjiev, N.: A Comparative Study: Service-Based Application Development by Ordinary End Users and IT Professionals. In: Di Nitto, E., Yahyapour, R. (eds.) ServiceWave 2010. LNCS, vol. 6481, pp. 163–174. Springer, Heidelberg (2010)
17. Hoyer, V., Fischer, M.: Market overview of enterprise mashup tools. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 708–721. Springer, Heidelberg (2008)
18. Zang, N., Rosson, M.B., Nasser, V.: Mashups: who? what? why? In Proceedings of CHI, Florence, Italy (2008)
19. Namoun, A., Nestler, T., De Angeli, A.: Service Composition for Non-Programmers: Prospects, Problems, and Design Recommendations. In: 8th IEEE European Conference on Web Services, Ayia Napa, Cyprus (2010)
20. Daniel, F., Koschmider, A., Nestler, T., Marcus, M., Namoun, A.: Toward Process Mashups: Key Ingredients and Open Research Challenges. In: 4th Workshop of Mashups 2010 (co-located with 8th IEEE European Conference on Web Services), Ayia Napa, Cyprus (2010)
21. Cao, J., Riche, Y., Wiedenbeck, S., Burnett, M., Grigoreauni, V.: End-User Mashup Programming: Through the Design Lens. In: Proceedings of CHI 2010, Georgia, USA, pp. 1009–1018 (2010)
22. Wong, J., Hong, J.I.: Making mashups with marmite: towards end user programming for the web. In: Proceedings of CHI 2007 (2007)
23. Pautasso, C., Frisoni, M.: The Mashup Atelier. In: Feuerlicht, G., Lamersdorf, W. (eds.) ICSOC 2008. LNCS, vol. 5472, pp. 155–165. Springer, Heidelberg (2009)
24. Jones, M., Churchil, E.: Conversations in developer communities: A preliminary analysis of the Yahoo! Pipes community. In: Proceedings of the 4th International Conference on Communities and Technologies, PA, USA (2009)

Designing a Framework for End User Applications

Yanbo Deng[1], Clare Churcher[1], Walt Abell[1], and John McCallum[2]

[1] Department of Applied Computing, Lincoln University, Lincoln 7647
Christchurch, New Zealand
Yanbo.Deng@lincolnuni.ac.nz, {Clare.Churcher,Walter.Abell}@lincoln.ac.nz
[2] Plant and Food Research, Private Bag 4704,
Christchurch, New Zealand
John.McCallum@plantandfood.co.nz

Abstract. Specialised end user developed database applications can often be designed and developed quickly and simply to meet specific needs. However these applications are often difficult to generalise or adapt when requirements inevitably change. In this paper we describe a framework that allows a basic data model to have several co-existing variations which will satisfy the requirements of different user groups in a common domain. A web service and development toolkits provide a simple programming interface for interacting with the database. User trials showed that end users were able to use the system to quickly adapt and create applications. The result allows the needs of several different groups of users to have their specialist needs managed within a single organisational database.

Keywords: End user development, Data management, Frameworks, Tool-kits.

1 Introduction/Background

There are a number of challenges involved with the development of applications in small organisations. Small organisations may not have the budget to buy commercial off-the-shelf software nor the expertise to create specialist software from scratch. Open source applications, which can be adapted to meet specific needs, are a possibility. However there is often a big learning curve for end users to understand how to customise and use these systems effectively.

In many organisations, end user developers (non professional developers) create small applications for their own use [1]. One of the major advantages is that these end user developed applications (EUDAs) can often be designed and developed quickly and simply to meet immediate needs [2-5].

Scientific domains are a good example of end user development activities. Many small-scale information systems (e.g. spreadsheets and small databases) have been developed by end users (such as scientists) to manage their specialist data for personal use. These applications are often difficult to generalise

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 67–75, 2011.

or adapt when requirements inevitably change. A critical issue is how existing EUDAs can be reused to meet other similar requirements in the same application domain.

We visited 10 research institutions in New Zealand, Australia and China to investigate issues of end user developed database systems. We found that different user groups in the same organisation might require similar but different data entities, attributes, relationships and constraints. These types of changes usually mean the data model needs to be amended and end users often find it simpler to develop separate systems to meet the new requirements. This eventually causes problems with long term data integration and maintenance.

Current software engineering techniques can assist professional developers to design flexible applications which can support changing requirements. Application frameworks are often used for flexible application development. Developers can extend functionality by sub-classing abstract classes from the framework and combining different types of objects together [6, 7]. A mature framework consists of many design patterns [8] that allow more complex functionality without necessary changing the structure of existing applications. A layered architecture is commonly used for information systems to improve application flexibility, which separates a mixture of presentation logic, application logic, and domain logic and persistence logic into independent layers [9]. This makes it easier to modify a layer without affecting other layers.

However, these traditional software engineering techniques usually require expertise beyond that of an end user developer. Many researchers have noted EUDAs require modifications and extensions to meet users further requirements. The Seeding, Evolutionary growth and Reseeding (SER) model [10] and Software Working Workshop (SWW) [11] suggest that end users should act as co-designers and work with IT professionals in order to evolve their applications for future needs. Component based approaches [12, 13] provide reusable components for end users to assemble to meet their specific needs in areas such as adding document searching functionality to applications.

One approach has been to give end users explicit tuition in how to effectively design databases. For example, a modified Entity-Relationship method (MER) was provided as a guide to tutors to help teach users how to model their data based on their specific problem domain [14]. Other researchers [15] have suggested that end users need to seek expert help for developing a comprehensive conceptual model in order to accurately reflect requirements. Methods were recommended to simplify the full conceptual model so that end users could confidently implement the model in a spreadsheet or database. Classes left out of the simplified model could be added as new tables in the future without affecting existing applications.

In this paper we describe a framework that will allow a basic data model to have several co-existing variations to satisfy the requirements of different groups in a common domain. We also provide tools to help users to extend the data model for new requirements and tools to help them develop new applications based on the updated model.

2 Case Study

Plant and Food Research (PFR) is a New Zealand government institute that carries out genetic research experiments on plants. There are many research groups who study different plants (e.g. onions, mushrooms and apples). One group of end users developed a small SQL server database to manage its onion samples and experimental information. An Excel based front end application was implemented (with expert assistance) to automate the experimental workflow and manage the database. For the original users (onion group), the application was successful in helping them efficiently access plant sample data, setup experiments and record experimental results. However, it was difficult to adapt the database and application to manage data for other research groups (mushrooms and apples) with similar but slightly different requirements.

These differences included:

- Additional or different data attributes
- Additional entities and different relationships among existing and new entities
- Different validation constraints and processes

Attempts to alter the original without affecting the existing applications proved too difficult and in the short term, separate databases were developed for the mushroom and apple user groups. This was clearly a sub-optimal solution in terms of long term management and data integration.

3 Framework Approach

Our approach is to recommend that a framework provider (a professional developer) create a minimal application framework to support different applications in the same application domain. With appropriate tools a framework manager can easily customise the framework for the specific needs of a range of end user developers.

Framework Rationale: The framework provider will work with end user developers to find a compromise “middle way” between a very specific design and a very generic approach in order to setup a framework. For example, PFR end user developers would like to create flexible applications that can be reused for onion, mushroom and apple data. An appropriate middle way point (shown in Fig. 1) should include all plants of possible interest but exclude more generic entities that are unlikely to be needed. A framework designed at the appropriate level will then only need slight changes to be adapted for the different

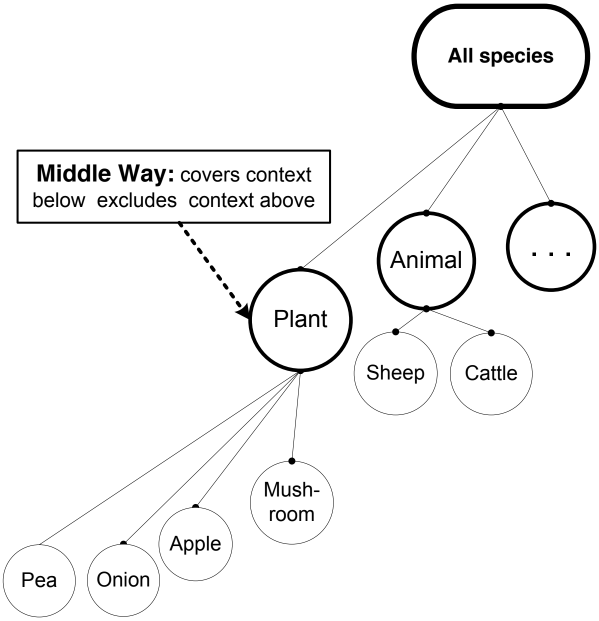


Fig. 1. Framework Rationale

specialist crops. In this case, a framework designed at this middle way point is likely to be suitable for a large number of laboratories dealing with plant genetic data.

Framework Design: Framework providers will design a domain specific framework based on the middle way. The framework will include abstract classes to represent collaborations and responsibility for domain objects/entities that are required as well as example concrete classes (e.g. for onion and apple samples). Lower level classes will inherit attributes, rules, and associations from the framework level and can be customised to meet specific needs. Fig. 2 shows the three level design for a database concerned with storing plant samples and their associated assays (experiments). Framework level abstract classes only have minimum attributes (e.g. Id and Name), minimum validation rules (such as sample id and name are required) and generic relationships between objects (e.g. many to many) for common situations.

Organisations will be able to customise organisational level abstract classes to include common attributes and validation rules for all user groups/departments within the particular organisation. The validation rules defined at this level will be inherited by all individual level concrete classes, which makes it possible to control and manage data from an organisational aspect. Using the tools we provide, the framework manager can create new concrete classes at the individual level to meet the requirements of different end user groups.

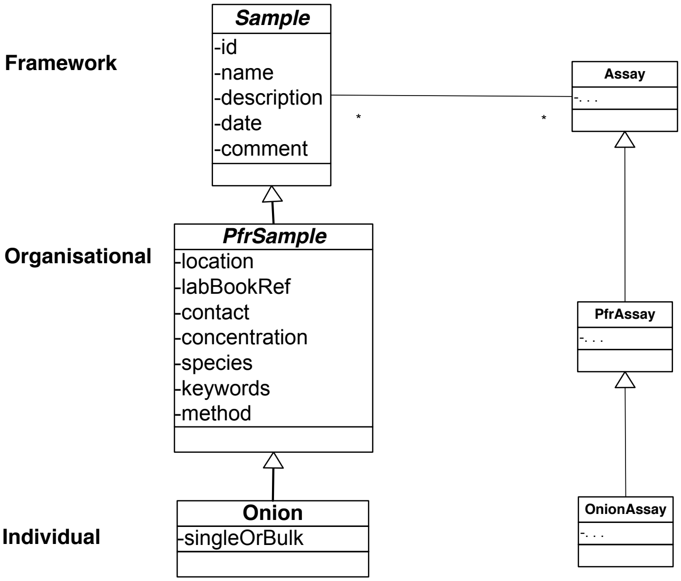


Fig. 2. Framework Design

System Overview: The overall system with the framework and associated services is summarised in Fig. 3.

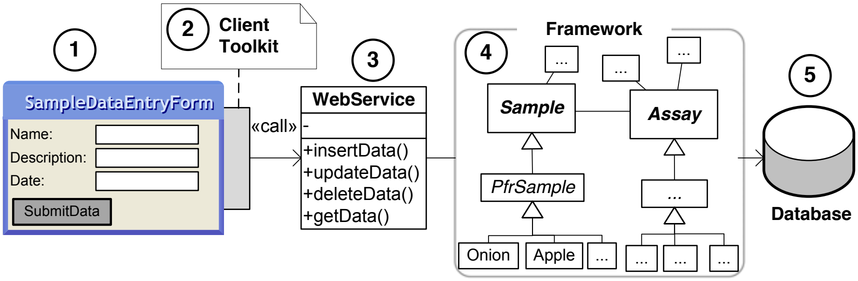


Fig. 3. System Overview

In Fig. 3 we see the framework (Part 4) with two concrete classes (Onion and Apple). Framework configuration tools are provided to help framework managers add new concrete classes (e.g. Mushroom).

A framework configuration file template is provided for defining individual level meta-data, such as class names, attributes and validation rules. The configuration tools parse the configuration files and generate Java class code to implement the framework.

The configuration tools also generate the corresponding database tables (Part 5). The one table per class mapping strategy [16, 17] is applied to convert the

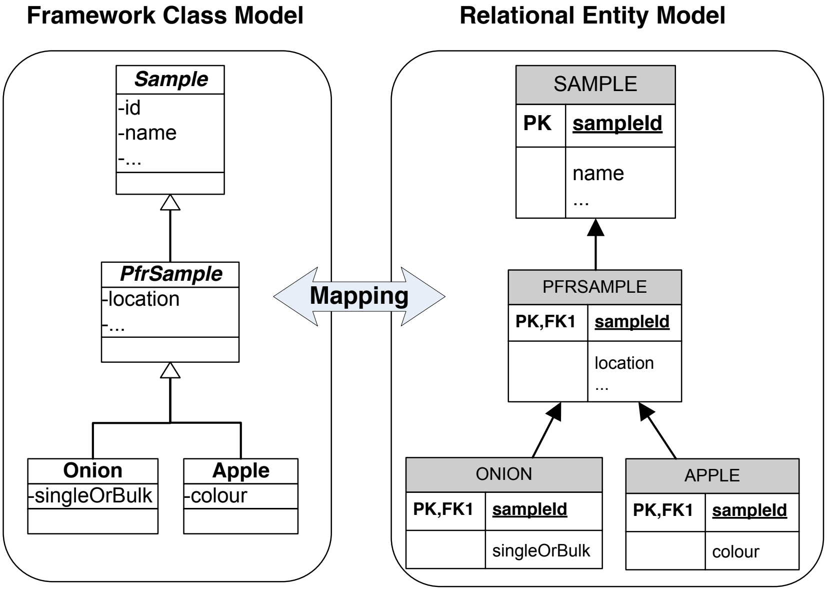


Fig. 4. Mapping framework classes to database tables

class model into a relational database model. Individual level tables have primary key associations to the framework and organisational level tables. This creates a one-to-one association between each level (shown in Fig. 4). This strategy makes it straightforward to map the framework classes to the database. It is also easy to add new individual classes without affecting the rest of the database.

Framework programming interfaces are implemented as a web service (Part 3 of Fig. 3) to supply essential data access methods (e.g. query and update) for end user developers. The web service coordinates validator and data access objects to manage validation and persistence logic. Hibernate [18] is used to provide the data access objects to map between class object attributes and the corresponding table columns. The Spring framework [19] and Apache CXF [20] are used to provide the SOAP web service and WSDL information.

A key goal was to minimize the effort of using the framework to create the client applications. Client application development toolkits (Part 2 of Fig. 3) are provided for a number of languages and platforms. The toolkits handle communication with the web service and provide a set of functions for inserting, retrieving and updating data. For example at PFR, a VBA client toolkit was developed in C#, and it can be invoked from any MS Office application.

End User Development: With the toolkits, end user developers are able to create their own user interface (Part 1 of Fig. 3). Essentially they need to identify the data they want stored in the database and then call the appropriate web service methods. To assist end users with these tasks, supporting tools are provided to generate example applications (e.g. for apples) based on the data definitions in the configuration files. Currently these tools generate simple Excel data entry applications which end user developers can customise.

4 Evaluation

The primary principle in framework design is that it should be generic enough to be reused for many different applications within a specified domain. We evaluated the framework reusability through user trials at several research organisations. Two framework trial tasks involved framework managers:

1. Extend the existing apple sample type with additional data and validations.
2. Add a new type of sample for oranges (including data and validations).

Five end user developers (occasional VBA developers) and three professional developers (.NET and PHP developers) acting as framework managers completed the user trial tasks. The participants all successfully used the supporting tools to create the configuration file (for updating and creating the concrete sample classes) and generate the database tables. The overall feedback was that the approach was a useful and efficient way to create and adapt databases for managing different types of samples.

A key goal of the toolkit is that it should enable end user developers to create applications with minimal effort. Two application development trial tasks were designed for end user developers:

1. Create a new Excel based data entry application (for loading orange sample data) using the provided tools.
2. Alter the data entry layout of the Excel application and modify the VBA code to correctly load the data.

Five end user developers who had no or very limited VBA programming experience (from three different research organisations) participated in this trial. For Task 1, the participants were able to successfully create an Orange data entry application within 10 minutes. In Task 2, the participants successfully customised the data entry application within 30 minutes. Some enthusiastic participants even adapted the application and framework for their own plant data. One end user developer said “it is much easier than creating and adapting data management applications from scratch”.

5 Discussion/Limitations

The framework transforms EUDAs from very specific approaches to a more generic approach. The framework provides a flexible software structure for end user developers to create similar applications to meet specific needs. By using subclassing the framework can support different specialist applications at the same time. Existing data and applications do not need to be altered as new specialised requirements develop.

The framework approach means all data can be saved in a single database system, thus solving the data integration problems caused by several independent systems. The supporting tools simplify the database development and management for framework managers. There is no need to write any code to create (and test) the individual concrete classes, database tables and web services. Framework managers only need to define configuration files and run the tools that automate the above processes. It can save a huge amount of time for database implementation.

More importantly, the client toolkit significantly minimizes end user development efforts. End user developers need only concentrate on developing the user interface and using the appropriate toolkit functions. Because the data management logic is encapsulated in the web services and client toolkit, development is greatly simplified for end user developers. The toolkits require little programming expertise, because example applications can be generated which are easily adapted by novice users for their specific needs.

Although the framework prototype has successfully solved some common EUDAs problems, application performance still needs to be explored in the future. The impact of using the one table per class mapping strategy and web services to provide the programming interface must be considered. We will look at improving performance by techniques such as caching frequent query results and compression of the web service XML data.

Another issue to explore is security. In the production stage, the entire organisational/enterprise data are stored in a central database, so authentication methods and user roles must be added to control and audit user access.

6 Conclusion

This paper discusses the benefits of using a framework approach to help end user developers create a variety of adaptable applications with differing requirements. We suggest that framework providers design a domain specific framework to provide a reusable and flexible structure for end user developers. Tools are provided to help framework managers add concrete classes to the framework to cater for specialised data requirements. A web service and client toolkits provide a simple programming interface for interacting with the database. Both framework managers and end user developers were able to use the supplied tools to quickly adapt the framework and customise applications. The result allows different groups of end user developers to have their specialist requirements met within a single organisation-wide database.

References

1. Rainer, R.K., Harrison, A.W.: Toward Development of the End User Computing Construct in a University Setting. Decision Sciences 24(6), 1187–1202 (1993)
2. Rivard, S., Huff, S.L.: An empirical study of users as application developers. Information & Management 8(2), 89–102 (1985)

3. Amoroso, D.L., Cheney, P.H.: Quality end user-developed applications: some essential ingredients. SIGMIS Database, 23(1), 1–11 (1992)
4. EUD-Net, End-User Development:Empowering people to flexibly employ advanced in-formation and communication technology (2003)
5. Lieberman, H., et al.: End-User Development: An Emerging Paradigm. In: End User Development, pp. 1–8. Springer, Netherlands (2006)
6. Oscar, N., Dennis, T. (eds.): Object-oriented software composition, p. 361. Prentice Hall International (UK) Ltd., Englewood Cliffs (1995)
7. Larman, C.: Applying UML: patterns: an introduction to object-oriented analysis and design, p. xix, 507. Prentice Hall PTR, Upper Saddle River (1998)
8. Gamma, E.: Design patterns: elements of reusable object-oriented software. Addison-Wesley professional computing series, p. xv, 395. Addison-Wesley, Reading (1995)
9. Fowler, M.: Analysis patterns: reusable object models, p. xxi, 357. Addison-Wesley, Menlo Park (1997)
10. Fischer, G., et al.: Meta-design: a manifesto for end-user development. Commun. ACM 47(9), 33–37 (2004)
11. Costabile, M.F., et al.: Building environments for end-user development and tailoring. In: Proceedings of 2003 IEEE Symposium on Human Centric Computing Languages and Environments (2003)
12. Anders, I.M., et al.: Component-based technologies for end-user development. Commun. ACM 47(9), 59–62 (2004)
13. Wulf, V., Pipek, V., Won, M.: Component-based tailorability: Enabling highly flexible software applications. International Journal of Human-Computer Studies 66(1), 1–22 (2008)
14. Ahrens, J.D., Sankar, C.S.: Tailoring Database Training for End Users. MIS Quarterly 17(4), 419–439 (1993)
15. Churcher, C., McLennan, T., McKinnon, A.: From conceptual model to end user implementation, p. 15. Applied Computing, Mathematics and Statistics Group, Lincoln University, Lincoln, N.Z (2001)
16. Fowler, M.: Patterns of enterprise application architecture. The Addison-Wesley signature series, p. xxiv, 533. Addison-Wesley, Boston (2003)
17. Churcher, C.: Beginning database design. The expert's voice, p. xxv, 240. Apress, Berkeley (2007)
18. Elliott, J., O'Brien, T., Fowler, R.: Harnessing Hibernate, p. xiv, 363. O'Reilly, Beijing (2008)
19. Walls, C., Breidenbach, R.: Spring in action, 2nd edn., p. xxxiv, 730. Manning, Greenwich (2008)
20. Hathi, R., Balani, N.: Design and implement POJO Web services using Spring and Apache CXF, Part 1: Introduction to Web services creation using CXF and Spring (2008), Available from http://www.ibm.com/developerworks/webservices/library/ws-pojo-springcxf/ (cited March 6, 2009)

From Human Crafters to Human Factors to Human Actors and Back Again: Bridging the Design Time – Use Time Divide

Monica Maceli and Michael E. Atwood

Drexel University, College of Information Science and Technology
3141 Chestnut St, Philadelphia, Pa 19104 USA
{Monica.Maceli,Atwood}@drexel.edu

Abstract. Meta-design theory emphasizes that future use can never be entirely anticipated at design time, as users shape their environments in response to emerging needs; systems should therefore be designed to adapt to future conditions in the hands of end users. For most of human history, all design was meta-design; designers were also users, and the environments of design and use were one and the same. Technology introduced a divide between the skilled producers and unskilled consumers of technology, and between design time and use time. In our increasingly complex technological environments, tomorrow‘s meta-designers must be able to anticipate the environment in which the end users will work in order to provide the flexibility for users to craft their tools. By exploring and projecting forward current trends in technology use, we have identified key principles for meta-designers and suggest that using them as design heuristics will aid meta-designers in crafting systems for future end-users.

Keywords: design, meta-design, design time, use time, heuristics, context.

1 Introduction

The interactive systems we use today were, almost exclusively, designed in the past. Similarly, the interactive systems we design today will be, almost exclusively, used in the future. Design and use are typically separated in time and, because they are separated in time, they occur in different contexts. Not knowing the context of use limits our abilities as designers to produce useful artifacts.

We intend the artifacts we design to aid people in solving their problems. But, since we do not fully know the context of use, we cannot fully understand those problems. As John Thackara noted “We’ve built a technology-focused society that is remarkable on means, but hazy about ends. It’s no longer clear to which questions all this stuff – tech – is an answer, or what value it adds to our lives” [1]. The need to be able to predict the future when designing was also the motivation for Rasmussen and his colleagues [2] to develop *cognitive work analysis*: “In fact, what we are looking for in our efforts to create a conceptual framework for description of tasks, activities, work domains, etc., is a model framework, a framework for *description* which can serve to compare results from analysis made in different contexts and domains, which

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 76–91, 2011.

can serve to *predict* what kind of phenomena are to be expected in one work situation, given results from studies in other environments” [2].

We need to predict the context of use because use time occurs after design time. At one time, however, design and use were closely entwined activities: *human crafters* designed tools through use and there was no distinctly separate design process. As technology advanced, industrialization introduced a divide between the goals of the setting of design (*design time*) and the setting of use (*use time*). Design time focused on experts creating a completed design artifact, while use time was oriented towards gradual user-driven evolution and change, responsive to environment and context. This tension between what could be accomplished at design time and what unpredictable situations the system would encounter during use has been an ongoing challenge to the evolving field of Human-Computer Interaction (HCI).

When environments of use were constrained to the workplace, our early HCI methodologies could strive to match known work tasks with suitable interfaces; this *human factors* approach focused on the line between man and machine and the interfaces that afford interactions between the two. In the 1990s, when technology moved into the home and into more complex environments of use and practice, HCI methodologies began to take a broader view of interaction, supporting *human actors* who controlled the technologies used in their daily lives [3]. Our current HCI methodologies and theories are largely oriented towards this “human actors” relationship between technology, users, and use.

However, recently developed technologies have allowed for complex and shifting contexts of use [4] as well as empowered users to design their own technological environments. Novel means of information and technology production (e.g. open source software development, mash-ups, commons-based peer production [5]) have radically changed the technological landscape. Users are again behaving as *human crafters* – controlling, designing, and developing not only their relationships with technology, but the very form and function of this technology.

As a result, our traditional HCI design time activities have become increasingly ill-suited to the unpredictability of real life use. As users become more empowered to design their own technology environments, HCI theory and methodology must shift as well to better support and shape these activities. In order to address these challenges, the conceptual framework of meta-design [6] suggests redirecting our attention towards bridging the differences between design time and use time through systems and techniques allowing for real-time co-design of systems. Users function as both consumers and designers of their environment and the boundaries of system design are extended from one original system to an ongoing co-design process between users and designers. In this continuously participatory design, users and designers contribute to an ongoing design discussion centered on modifying the system in use.

It remains a challenge in information systems to orient design time conversations towards future use time and to understand the role of participation in such a process. As meta-design theory reinforces, future use can never be entirely anticipated, yet some work must happen at design time. This raises the key questions: what techniques can help anticipate some use time possibilities at design time? And how can designers and users communicate around the inevitable and unanticipatable future changes that will arise over the life of the system?

We begin with the following observations. First, the world is changing, or at least how we view the world is changing, and this will necessitate changes in the way we think about design. Second, while we cannot predict with certainty how the world will change, there are some changes that are becoming apparent. Technology is used increasingly for communication; technology supports people with similar interests in connecting; any technology is rarely used in isolation and almost always used with other devices; system users will continue to have little, if any, patience for learning how to use a system; the most successful technologies are those that can be personalized. Third, the distinction between *designer* and *user*, which once was nonexistent, will once again become blurred as people take increasing responsibility for the design of the systems they use.

2 A Brief History of the Design of Interactive Systems

At one point, we all agreed on what *design* is and it was very easy to do! But, that was long ago, and we are very far past the point. Below, we present a *brief* history of design as it relates to interactive systems. Our goal is not to present a complete history of design and design science, but to present the highlights that relate to the design of interactive systems.

2.1 In the Beginning: Human Crafters

People have practiced design for roughly 2.5 million years! As Mayall describes in *Principles of Design* [7], design, viewed as the creation of artifacts used to achieve some goal, traces back to the development of stone tools. No formal descriptions of design methods or design theories have survived from these early design efforts, and we doubt that they existed.

Mayall [7] notes that early design was driven by the belief that *new* is *better* and that *technology* is *good*. The earliest designers did not worry about unintended effects that their artifacts might have on individuals or society, and they did not spend a great deal of time focusing on user needs. New is better and technology is good, and these stone tools were clearly better than what preceded them. That groups of people might work together, that the context of use might influence design, that people might have need for multiple tools at one time or might use one tool in multiple ways were not factors that needed serious consideration.

Design in this phase was a common human activity, describing a general process of identifying an environmental misfit and moving towards a more desirable state in pursuit of some goal [8]. Design time and use time were once intermixed; nearly anyone could do it or be witness to it, tools were designed through use, there was no separate design process, and there were certainly no experts conducting the "science of design."

The artifact‘s inability to perform its function (on any number of dimensions) was the motivating factor for design changes. As Petroski claims, "one thing follows from another through successive changes, all intended to be for the better" [9]. This small-scale, often trial-and-error, iterative movement away from misfit towards a better solution characterizes the vast majority of historical acts of design. Even moving into the industrial revolution, many of the enabling technologies arose from the work of practical inventors and entrepreneurs, often with little education and theoretical knowledge (or need for such knowledge) in the sciences [10].

However, with the industrial revolution came a huge increase in machine based manufacturing, and work practices in turn began to focus on efficient separation of labor. It is around this time that the focus of design shifted from the tools themselves, to the creation process. During this time period, machinery was operated by humans in the context of work and rarely, if at all, by choice. Attempts to improve this "non-discretionary use" [11] focused on increasing the speed, safety and efficiency of known and constrained work tasks. Industrialization introduced and reinforced a divide between the producer and consumer, and "high tech scribes" (those fortunate enough to have system building skills) begin to monopolize both the design and the solution of tools and systems [12].

2.2 Fitting Man to Machine: Human Factors

In the 1950s, things begin to change. We will not attempt to document these changes and their causes here other than to note that there were significant advances in technology and significant changes in social structure during this time. Dreyfuss [13] was one of the first, if not the first, designer to focus on that fact that the goal of design was to fashion artifacts that people could use. Very early in the formalization of design methods, he noted that "We bear in mind that the object being worked on is going to be ridden in, sat upon, looked at, talked into, activated, operated, or in some other way used by people individually or en masse" [13]. Although a focus on people may sound commonplace today, it was revolutionary in the 1950s.

By the 1960s, new technologies and new uses for systems had reached the point where it was becoming clear that something needed to change about the way we thought about design. Christopher Alexander acknowledges in *Notes on the Synthesis of Form* [14] that many design problems were reaching "insoluble levels of complexity." Design problems that once looked simple and that had simple solutions now looked more complex and new, more formal approaches to design were proposed.

By the 1980s widespread use of personal computers sparked tremendous interest in what would emerge as the human-computer interaction community. This work was very much in what Bødker [4] calls *first wave* HCI. Usability was the primary focus of both researchers and practitioners. And, usability was understood to involve one person interacting with one system to do one thing. Interfaces provided the means for man and machine to interact during work tasks. The field of human factors emerged from a desire to increase workplace performance and decrease errors on the part of the human operators. Experts in the growing field of human factors took up many of these interface concerns and published some of the first articles in the field of human-computer interaction [eg. 15, 16]. This early research and many human factors handbook guidelines [eg. 17] focused on computer systems tailored to fit human operators in the context of required work. These handbooks brought a formal focus towards the process of design and towards considering users as the ultimate consumer of the system, albeit a narrow work-centric view of the user as an element of the system whose limitations must be designed around.

However, difficulties operating the computing systems of the time encouraged the involvement of cognitive psychology, in hopes of leveraging this body of knowledge to inform new and usable systems [18]. As psychology researchers became involved this dictated a focus on human cognition and behavior, in attempts to achieve better "cognitive coupling" between humans and computers [3]. In 1974, Newell, Card, and Moran undertook a project to tie existing psychological theory and data to human-computer interaction, constructing the Model Human Processor and GOMS methodology for modeling human cognition and tasks [19]. This framework allowed experts to conduct a quantitative prediction of user performance, and could be used to evaluate existing and proposed interfaces.

These developments mark a concerted effort to apply existing psychological theory to HCI, with hopes of arriving at one, generalizable model of human cognition that would be relevant to design. Models of human cognition were constructed and the affordances and constraints of the technology artifacts were analyzed. In the tightly constrained workplace environment, these methods could yield successful results: in better fitting man to machine, avoiding errors, and improving performance. The challenge of designing for an unpredictable world was suppressed when routine work tasks in easily observable environments were the focus of design and evaluation activities.

The cognitive models that evolved during this period (e.g. Model Human Processor, GOMS, directness, theory of action) remained the dominant approach for many years. However, they eventually gave way to a more situated, evolving view of human interaction, as computers moved from the workplace, into the more complex environments of our daily lives and our homes – all complex use time scenarios not easy or even possible to model.

2.3 Users as Active Agents: Human Actors

In *second wave* HCI, which Bannon [3] described as the move from human factors to human actors, the focus shifted to *groups of people*, typically in established communities of practice, working toward *common goals* with a *collection* of applications. Communication and collaboration became key topics for research and practice. While design theorists and HCI professionals were largely separate communities at this time, they did share a focus on collaboration. For example, Rittel [20] proposed formalisms to help resolve what he saw as the symmetry of ignorance in urban planning; Ehn [21] focused on ways to involve users in design efforts; and Rasmussen et al [22] focused on socio-technical systems such as nuclear power plants, where collaboration between people is key.

Additionally, a strong Scandinavian tradition and body of research had been building since the 1970s, with user participation as the focal point of design. A cultural emphasis on collective resources and multiple union-driven projects had dictated the inclusion of workers in the design and development of workplace computer applications. In a Scandinavian collection of writings oriented around cooperative system design and written in 1991, Liam Bannon's article "From Human Factors to Human Actors" [3] marked a transition away from what Suzanne Bødker [4] terms first wave HCI.

Reflecting upon current conceptions of HCI, Bannon cited a disappointing lack of contribution to practice from the predominant cognitive psychology paradigm, and that such a theoretical emphasis was of little use to designers with immediate, real world problems to solve. Bannon called for a re-orientation towards usability: which would allow for "quick and dirty" techniques to help designers understand a product‘s utility and usability, within an iterative process that involved users as active agents in the process. Along with the methodological implications, Bannon argued for a theoretical move from product to process, from individuals to groups, from analysis to design, and from the laboratory to the workplace.

In the same collection, Hendersen and Kyng put forth several ideas towards a future of *designing in use* [23]. They described the key challenges of design as arising from the differences between the setting of design and the ultimate setting of use. Design, in their view, should address: changing situations of use, the complexity and unpredictability of the real world, and the need to design for many different situations of use. This would be achieved through designing for tailorability, or creating the artifact with change in mind.

These perspectives advocated for moving design time closer to an increasingly complex use time; theoretical frameworks focusing on social, emergent interaction "in the wild" were developing to better understand this complex use time. In 1987, Lucy Suchman's published dissertation, *Plans and Situated Actions* [24], proposed an alternate view of human-computer interaction in her theory of situated action. Informed by a social science and ecological psychology perspective, Suchman refutes the prevailing cognitive psychology belief that human behavior is purposeful and planned, rather she argued that human action is more shaped by the immediate, surrounding contextual environment and circumstances. With newly minted theories and methodologies, the majority of them qualitative in nature (e.g. situated action, distributed cognition, Activity Theory), HCI gained new ways of describing and understanding use and context [25] as it related to real life activities.

However, there was often little connection between these theories and predictive design methodologies for practitioners to employ. The early 1990s were characterized by an increased interest in cognitive modeling approaches (e.g. cognitive walkthrough, heuristic evaluation) viewed as more applicable to practice and in conducting empirical experiments with real users, under the umbrella term of "usability engineering". These usability techniques gained popularity quickly as they were pragmatic, prescriptive, and relatively easily communicated to design practitioners (who, increasingly, were not the theorist and psychologists of the early days of HCI, but rather commercial designers outside academia).

However, the rush to create techniques that were applicable to real life design scenarios left some researchers questioning their effectiveness. In 1998, Gray and Salzman [26] in their critique of usability studies at the time, provided an important reminder that the goal of usability is to improve the "outcomes of interest", which could widely vary depending on the type of product being developed. And they pointed out a lack of empirical work proving the connection between improved usability and improved usefulness of the eventual product (which remains an open issue to this day).

2.4 A Return to Designing in Use: Human Crafters

In Bødker's [4] *third wave HCI*, "the use context and application types are broadened, and intermixed, relative to the focus of the second wave on work. Technology spreads from the workplace to our homes and everyday lives and culture." She continues to note three challenges – (1) people need to be involved in design, not just as workers, but as someone who brings their entire life experience into the design, (2) this will necessitate a change in the way we design and prototype, and (3) we need to move away from end-user programming in isolation to configurations involving multiple people and multiple systems. In the section immediately below, we will consider how we might address these challenges. We have described the evolution of HCI as shaped by a changing relationship between design time and use time and we note, in both literature and practice, a return to *human crafters*.

Like the earlier stage of human crafters, the goal is to reduce the separation between design time and use time. Unlike that earlier phase, however, we are designing for a world which is increasingly technological. Evaluating the usability of interfaces and observing real-world use, although valuable techniques, focus on the "immediate needs of the immediate users" [27]. The unanticipated futures of the technology when released in the world remained problematic; the new interactions it would facilitate and the necessary changes and modifications that would arise through use were unknown.

Henderson and Kyng's vision of "designing in use" was not the first to approach the problem of designing for unpredictable futures. These ideas were highlighted in earlier influential works: notably Christopher Alexander's vision of an "unselfconscious culture of design" [14] where users had the skills and confidence to tailor their environment, and Ivan Illich's concept of convivial technology tools [28] that would empower people to conduct creative and autonomous actions. These largely theoretical works described a fundamentally different culture of design, one which introduced complex questions around the goal of allowing and encouraging users to act as (and with) designers.

Although these ideas had been around for many years, the challenge of continuously designing in use was rapidly becoming unavoidable. In 2006, John Thackara, in writings centered around sustainable and complex design practices notes that "against this backdrop of situations in which systems don't stop changing, the idea of a self-contained design project—of 'signing off' on a design when it is finished—makes no sense" [1]. Bødker's suggested "3rd wave" goal of "re-configurability in the hands of networks of human users" [4] was becoming not just a vision, but a requirement.

However, many open questions exist around how to practically support users during the process of designing in use. Anticipatory techniques explored what could be done at design time (often by designers alone) to endow the system with the properties to be flexible in use. Architect Stewart Brand's process of scenario-buffered design [27] encouraged users and designers to strategize around potential future uses for the building and space, yielding a final design that could respond well to multiple futures, not just the "official future" that was initially envisioned.

In the field of information systems, Gerhard Fischer's research has explored how systems can be user modifiable such that they might be designed in use. Early work focused on building knowledge-based design environments [29] (or DODEs) which provide a constantly evolving space in which users can create, reflect, and shape the system. Fischer‘s Seeding, Evolutionary Growth, and Reseeding (SER) Model [30] attempted to address the changing nature of use as the system evolved. In this model, a participatory design (or co-design) process between environment developers and domain designers yielded a “seed” within which as much information as possible is designed. This seeded environment is then used by domain designers on real projects, allowing for evolutionary growth through system use, until it becomes unwieldy and the reseeding process, organizes, generalizes, and formalizes.

Fischer, in more recent work [eg. 6, 31], addresses these issues and endeavors to take HCI beyond the limitations of participatory design methods and towards a future of user-centered development or *meta-design*. Taking on many of the challenges inherent in Bødker‘s “third wave” of HCI, meta-design describes a future state of design consisting of open systems that evolve during use, with design activities redistributed across time and levels of interaction with the environment. The framework emphasizes that the design of socio-technical systems must support flexible and evolving systems, that are not (and cannot be) completely designed before use, and that evolve in the hands of their users.

In attempting to shift design activities from “design time” towards “use time”, Fischer and Giaccardi [6] suggest a number of dimensions that come into focus as distinctly different from traditional design concerns: process over object, co-creation over autonomous creation, seeding over complete system, construction over representation, and exceptions and negotiations over guidelines and rules, among others. Further diverging from earlier HCI techniques, meta-design suggests that designers must give up control to users and that users may play the role of consumers or designers depending on context [31].

In practice, true meta-design methodologies that can be generalized across domains are not forthcoming. An attempt has been made to extract useful principles from identifying and exploring existing models of success, such as open source software development as a model for distributed cooperative work, as well as to explore and explain, from a meta-design perspective, domains such as interactive art and learning communities. The vision of meta-design is far-reaching and deeply embedded in both social and technical issues, raising questions as to the nature and motivation of participation, the evolving practices of communities, and the limitations and capabilities of our technologies, among others.

Furthermore, highly influential technologies and infrastructures, such as the Internet, have given rise to new modes of information production with little direct involvement from HCI theory or practice. Yochai Benkler, in *The Wealth of Networks* [5], describes how new information production models have emerged in the increasingly ubiquitously networked environment, as the physical costs of information production declines through widespread computer networks. Commons-based peer production [5] describes the efforts of thousands of individual volunteers, exploiting computer infrastructures such as the Internet, in order to collaborate, organize, and cooperate on immensely powerful, creative, and popular information products and systems (e.g. Wikipedia, Apache web server). Few of these developments involve formal usability methods or qualitative techniques; these projects arise for and by users’ own actions, playing the fluctuating role of both designers and users.

3 Designing for Design in Use

Designing in use supports users acting as designers, as well as systems that must continuously evolve to conform to future, unpredictable needs. It requires design time thought to be focused away from immediate needs and towards common emergent behaviors that users engage in over time. These behaviors center around: *connecting* – to people with similar interests or needs, *having conversations* – in real-time across space and time, *combining* – the system with other tools and systems they use, *getting up to speed quickly* – so undue time is not spent learning the system, and *tailoring* – such that the system is molded to their personal needs. These behaviors originate from our growing understanding of real world environments of use, informed both by theory and practice, and the many perspectives and complexities of this use time. Looking at how interactive systems are currently used suggests that these behaviors are already beginning to emerge.

3.1 Understanding the Complexities of Use Time

By the late 1990s, the field of HCI had moved away from the basic science laboratory and firmly into real life practice, but, as Gray and Salzman [26] illustrated, there was still relatively little understanding of how to improve system design in a measurable, useful way. Donald Schön [32] notes that the problems with the highest relevance and social importance may be those in which it is hardest to obtain technical rigor; the field of HCI struggled with this issue as it moved away from the laboratory.

Gibson's ecological psychology [33], Vincente and Rasmussen's cognitive work analysis [34], and Lucy Suchman's situated action [24] were developing theories suggesting a shift in design towards the environment and setting of use. Theories from other nearby fields were being incorporated into HCI as well, as the movement away from cognitive psychology continued: Activity Theory emerged from Soviet psychology [35], external cognition from Scaife and Rogers [36], distributed cognition from the anthropological work of Ed Hutchins [37], and Card and Pirolli‘s information foraging [38]. These newer approaches yielded rich, explanatory descriptions, with a focus on providing formative, generative, and analytic frameworks [18]. While they are impressive as theoretical frameworks, they have not evolved into prescriptive design methods.

Brown [39] suggested a "pioneering" research approach with new organizational principles devoted to learning from local, innovative work practices and moving towards the co-production of innovation with the customer. Although participatory design had begun to pave the way for including customers in the process of design, Brown took these concepts further, suggesting a future in which information technology is rendered invisible: "information technology will become a kind of generic entity, almost like clay. And the 'product' will not exist until it enters a specific situation, where vendor and customer will mold it to the work practices of the customer organization" [39].

What Brown envisions is a *participatory co-design process* in which designers envision the future context of users and shape systems that users can further refine, likely with additional iterations as designers learn from user actions and further shape their designs.

3.2 Is Participatory Co-Design Possible?

The ideas of co-production and co-design are gaining momentum in participatory design practices wherein users and designers work together to envision future contexts of use. However, as earlier researchers had cautioned [eg. 23, 40] *design time* processes still could not capture all the complexity and possibilities of real-life *use time*. As mentioned earlier, Fischer et al.'s [29, 41] three-phase model of Seeding, Evolutionary Growth, and Reseeding (SER), attempts to address the changing nature of use as the system evolves. In this model, a participatory design (or *co-design*) process between environment developers and domain designers yielded a "seed" within which as much information as possible is designed. The "seed" can never contain all of the required information, as the unpredictable use time will dictate what additional information is of relevance. This seeded environment is then used by domain designers on real projects, allowing for evolutionary growth through system use, re-seeding collaboratively when evolutionary growth stops proceeding smoothly.

Fischer et al.'s SER model moves design closer to the setting of use, allowing the system to be shaped and molded directly by end users. However, this model is heavily based on participatory design techniques, which largely focus on activities taking place at design time. And the counterproductive divide between programmers and users remains problematic: the system does not achieve infinite evolutionary ability before becoming unwieldy and requiring developers to step in. This research does begin to directly address the balance between system flexibility, or the ability to evolve through use, and system rigidity, or designing enough to minimally support use.

Information systems are not the only designed artifacts to struggle with the dilemma of designing for an unknown future of use; the architect Stewart Brand proposed a process of *scenario buffered design* in which building designers strategize around potential future uses, seeking to avoid "making the building all too optimal for the present and maladaptive for the future" [27]. He proposed the unit of analysis of design to be not simply the building itself but rather *the use of the building throughout time*. In this "future oriented process of design and decision" [27], he advocated for a participatory process focusing on scenario creation exercises, beginning with identifying the driving forces that will shape the future and extending these into the "official future" and a number of implausible, but possible, other scenarios. In this tactic, an unpredictable future is assumed, but in theorizing around divergent scenarios, adaptability and flexibility are naturally built into the design. Similar to Fischer, Brand suggests shifting design power to the user, in creating buildings that afford easy servicing by the users and opportunities to develop a hands-on relationship with "their" space.

These goals, of continuous and participatory system evolution, have emerged in models of software production such as the open-source movement. Based on "voluntary contributions and ubiquitous recursive sharing; on small incremental improvements to a project by widely dispersed people…" [5], such systems illustrate the possibilities for future end-user development. However, most contributors involved in the development of open-source systems are, by necessity, highly skilled in such activities, leaving minimal opportunities for contributions by novice users.

3.3 Principles for Designing in Use

These highly complex environments of use, consisting of rapidly evolving technologies and new means of information production, require a new focus for design activities. We would like to adopt Brand's approach by shifting design power to the user. The challenge that remains is that we must learn how to think about designing for such a world. That is, we must predict the future context in which the users will work. We suggest the series of principles that follow – aimed at orienting design time activities towards future use, as well as providing a frame for users and designers to communicate changes across the entire life of the system. These principles are derived from consolidating the broad literature on participatory co-design.

Table 1. Principles for Designing in Use

People like to use systems where they can:

1. **Connect** with other people with similar needs and interests, both nearby and far away.
2. **Reach out** and converse with other people in real-time, while they are using the system.
3. **Combine** it with other tools and systems they use regularly.
4. **Begin using it quickly**, without a lot of help or instruction.
5. **Tailor** it to their personalized needs.

The rationale behind the inclusion of each guideline is described below. We have also briefly noted trends in real-world system use that indicate users are extending existing systems in these directions.

Guideline 1. Connect with other people with similar needs and interests, both nearby and far away.

John Thackara's [1] series of design frameworks for complex worlds emphasizes the increasing importance of systems that allow people to connect and communicate both locally and across the boundaries of time and space. Technology systems provide a valuable means of connecting, building, and extending the communities of practice that support users in many aspects of their daily lives. This guideline intends to encourage these possibilities by focusing designers on how users can use the system to connect to similar people, and how they might attempt to extend the system in this fashion. The massive popularity of social networking tools (eg. Myspace, Facebook) emphasizes how powerful the need to connect with other people is and how users seek to make such connections across nearly every dimension of their lives' (work, education, health, dating, etc.) even if systems are not intentionally designed for these purposes.

Guideline 2. Reach out and converse with other people in real-time, while they are using the system.

Research prior to meta-design has explored modifiable systems that allow for reflective use-time conversations to occur, between designers and users [eg. 42]. This guideline seeks to emphasize how users can have live experiences and conversation with other people within, or around, the system. This may be with other users, with designers, or with knowledgeable users acting as designers. And, more generally, people use their social networks to accomplish their goals and answer questions, even if it means ignoring "formal" channels [eg. 24]. Research in recent years has explored the emergent use of chat and microblogging tools, such as Twitter, to facilitate backchannel conversations during conference presentations [eg. 43], as well as in many other domains, such as collaborative learning.

Guideline 3. Combine it with other tools and systems they use regularly.

The new (or redesigned) system may be only one of several tools and systems they use on a daily basis or even at the same time. As Bødker emphasized, technology configurations will increasingly involve multiple people and multiple systems [4]. Designing for these complex environments is a challenge facing HCI today and many theoretical frameworks (e.g. distributed cognition [37]) describe the intensely combinatory and situated nature of real life use. While designers can never anticipate exactly how their system might be used, they can view it as only one piece of a larger, evolving puzzle and not assume it to be a discrete system with 100% of the user's focus. And as Thackara points out, this focus, to the surrounding edge and combinatory effects, may spark new ideas [1]. This behavior is currently evident in the increasing popularity of web application "mashups", where users can combine disparate data sources and programming interfaces to create novel tools; this trend is reflected in the emerging literature studying this phenomenon [eg. 44].

Guideline 4. Begin using it quickly, without a lot of help or instruction.

Alexander's unselfconscious culture of design [14] requires systems users can understand relatively quickly and then contribute to confidently. This breaks down mental and physical barriers that prevent users from understanding the space or system well enough to have opinions and take actions to modify it. The goal of this design exercise is not to overload users with a multitude of features; this guideline is oriented towards envisioning ways in which novice users could begin using systems quickly and confidently, potentially becoming empowered to act as designers. There are many open questions around what motivates users to contribute to design activities; breaking down obvious barriers to use can encourage users acting as designers. The continuing attention paid to system *usability* in both academia and industry over the past few decades underlies the users' need for effective, usable systems.

Guideline 5. Adapt it to their personalized needs.

Henderson and Kyng's [23] early writings on designing in use identified tailorability as essential to systems supporting users acting as designers. There are many ways in which systems can be tailorable or adaptable: the system may tailor itself to the particular individual's needs automatically or through the user's tailoring actions. Successful systems, at this stage of technological development and users expectations, will likely all require some level of personalization and tailoring. It is the intent of this guideline to bring these needs to the forefront of design discussions and decisions. Considering tailorable aspects can put the necessary tools into the hands of the users if (and when) the need for future modifications arise. A recent example of the necessity of supporting such behavior is exemplified by the Apple iPhone. Though these devices are officially designed to be exactly the same, users can extensively personalize and adapt the device to their needs; the popularity of the AppStore, which shares and sells tools designed by and for users, emphasizes the natural tendency towards continuing design in use.

4 Conclusion

In The Invisible Computer [45], Donald Norman addressed the fundamental (and in his view – inescapable) complexity of computing devices; the more functionality computers were tasked with, then the more complicated and hard to use they became. No amount of designing could remedy this situation and the relentlessly evolving business model of the PC industry forced continuous change. Similarly, Bill Buxton [46] notes that while human capacity remains constant, technology offers continuous growth in functionality, leading to a future in which technology passes the "complexity barrier" and exceeds human capabilities.

Using the evolution of motors found in common household appliances as a model, Norman describes how motors became "embedded within these specialized tools and appliances so that the user sees a task-specific tool, not the technology of motors" [45]. He advocated for a movement towards information appliances which perform specific information-related activities, fitting to the task exactly, and provide universal communication and sharing. In 2001, Buxton [46] extended these ideas, describing the properties and tradeoffs of "super appliances" designed with attention to physical and social context, as well as relative to other information appliances. And Brown suggested a future in which information technology is rendered invisible: "information technology will become a kind of generic entity, almost like clay. And the 'product' will not exist until it enters a specific situation, where vendor and customer will mold it to the work practices of the customer organization" [39].

In today‘s information environment, the nature of technology tools has evolved towards Norman‘s vision of information appliances; in a typical day we encounter and interact with multiple computer-processor-powered artifacts, many of them "invisible". The popularity of the ubiquitous computing research theme continues to build in tandem with such tools. This multiplicity of interaction is a key challenge defining the 3rd wave of HCI, as described by Bødker [4]. However, a rift between HCI application and theory has emerged as a result of these advances; technologies have begun to enable quick and easy changes pieced-together during use, which is not reflected in our design techniques.

HCI still operates within a traditional model where design time and use time are separated. Even though this distance may be minimized, such as in participatory and scenario design techniques, it still exists and it still divorces the resulting tools from the complexities of real world use; in the words of Stewart Brand it “over-responds to the needs of the immediate needs of the immediate users” [27]. There is still a “time out”, however brief, when technology tools go back into the hands of the designers to be modified. Approaches such as Fischer and Giaccardi‘s meta-design attempt to confront this distance directly, but need further exploration to provide generalizable design methods.

These meta-design methods must support designing interactive systems that mold to both our present and our future. For the design of interactive systems, the future and the past are both similar and dissimilar. They are dissimilar in that the future will find people surrounded by technology (indeed, this already becoming true of our present!). These ubiquitous technologies will help people connect and converse with other people and these technologies will be used in such a way that they are integrated and combined with other technologies. Despite the power and ubiquity of these technologies, people will have little, if any, tolerance for learning to use them. What we say here is true today and these trends will only increase in the future.

While dissimilar in some ways, the past and present of design are similar in one regard. For millions of years of human history, there was no separation between design time and use time or between the design environment and the use environment. For most of human history, all design was meta-design; the people with the need crafted the tools to meet that need.

But, future meta-design is unlike past meta-design in one important regard, while the power to craft tools, in both cases, rests with the end user, for the present and future, we still need designers with specialized skills. Meta-design of stone axes is different from meta-design of interactive systems. In the past, the skills needed to design were possessed by everyone. Today and tomorrow, we need designers to first craft the tools which end users will craft later.

Tomorrow‘s meta-designers must be able to anticipate the environment in which the end users will work in order to provide the flexibility for users to craft their tools. Anticipating that future does not require a crystal ball, rather, looking at current trends in technology use and projecting them forward reveals key principles to meta-designers. We have outlined some of these key principles in this paper. We suggest that using them as design heuristics will aid meta-designers in crafting systems for future end-users.

References

1. Thackara, J.: In the Bubble: Designing in a Complex World. MIT Press, Cambridge (2005)
2. Rasmussen, J., Pejtersen, A.M., Schmidt, K.: Taxonomy for cognitive work analysis. Risø National Laboratory (1990)

3. Bannon, L.J.: From human factors to human actors: the role of psychology and human-computer interaction studies in system design. In: Greenbaum, J., Kyng, M. (eds.) Design At Work: Cooperative Design of Computer Systems, pp. 25–44. L. Erlbaum Associates, Hillsdale (1991)
4. Bødker, S.: When second wave HCI meets third wave challenges. In: Proceedings of the 4th Nordic Conference on Human-Computer Interaction: Changing Roles. ACM, Oslo (2006)
5. Benkler, Y.: The Wealth of Networks. Yale University Press, New Haven (2006)
6. Fischer, G., Giaccardi, E.: Meta-Design: a framework for the future of end user development. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development, pp. 427–458. Springer, Dordrecht (2006)
7. Mayall, W.H.: Principles in Design. Design Council, London (1979)
8. Alexander, C.: The Timeless Way of Building. Oxford University Press, New York (1979)
9. Petroski, H.: The Evolution of Useful Things. Knopf, New York (1992)
10. Stokes, D.E.: Pasteur's Quadrant: Basic Science and Technological Innovation. Brookings Institution Press, Washington, D.C (1997)
11. Grudin, J.: Three faces of human-computer interaction. IEEE Annals of the History of Computing 27, 46–62 (2005)
12. Fischer, G.: Computational Literacy and Fluency: Being Independent of High-Tech Scribes. In: Engel, J., Vogel, R., Wessolowski, S. (eds.) Strukturieren - Modellieren - Kommunizieren. Leitbild mathematischer und informatischer Aktivitäten, pp. 217–230. Hildesheim, Franzbecker (2005)
13. Dreyfuss, H.: Designing for People. Simon and Schuster (1955)
14. Alexander, C.: Notes on the synthesis of form. Harvard University Press, Cambridge (1964)
15. Shackel, B.: Ergonomics for a Computer. Design 120, 36–39 (1959)
16. Smith, S.L.: Man–Computer Information Transfer. In: Howard, J.H. (ed.) Electronic Information Display Systems, pp. 284–299. Spartan Books, Washington (1963)
17. Woodson, W.E.: Human Engineering Guide for Equipment Designers. University of California Press (1954)
18. Rogers, Y.: New Theoretical Approaches for Human-Computer Interaction. Annual Review of Information Science and Technology 38, 87–143 (2004)
19. Card, S.K., Moran, T.P., Newell, A.: The Psychology of Human-Computer Interaction. Lawrence Erlbaum Associates, Hillsdale (1983)
20. Rittel, H.: Second-Generation Design Methods. In: Cross, N. (ed.) Developments in Design Methodology, pp. 317–327. John Wiley & Sons, New York (1984)
21. Ehn, P.: Work-oriented design of computer artifacts. Erlbaum, Hillsdale (1989)
22. Rasmussen, J., Pejtersen, A.M., Goodstein, L.P.: Cognitive systems engineering. Wiley, New York (1994)
23. Henderson, A., Kyng, M.: There's No Place Like Home: Continuing Design in Use. In: Greenbaum, J., Kyng, M. (eds.) Design At Work: Cooperative Design of Computer Systems, pp. 219–240. L. Erlbaum Associates, Hillsdale (1991)
24. Suchman, L.: Plans and Situated Actions: The Problem of Human-Machine Communication. Cambridge University Press, Cambridge (1987)
25. Dourish, P.: What we talk about when we talk about context. Personal and Ubiquitous Computing 8, 19–30 (2004)
26. Gray, W.D., Salzman, M.C.: Damaged merchandise? A review of experiments that compare usability evaluation methods. Human Computer Interaction 13, 203–261 (1998)
27. Brand, S.: How Buildings Learn. Viking, New York (1994)

28. Illich, I.: Tools for Conviviality. Harper & Row Publishers, New York (1973)
29. Fischer, G.: Domain-Oriented Design Environments. Automated Software Engineering 1, 177–203 (1994)
30. Fischer, G., McCall, R., Ostwald, J., Reeves, B., Shipman, F.: Seeding, evolutionary growth and reseeding: supporting the incremental development of design environments. In: Conference Seeding, Evolutionary Growth and Reseeding: Supporting the Incremental Development of Design Environments, pp. 292–298 (Year)
31. Fischer, G.: Meta-Design: Expanding Boundaries and Redistributing Control in Design. In: Conference Meta-Design: Expanding Boundaries and Redistributing Control in Design, pp. 193–206 (Year)
32. Schön, D.A.: The reflective practitioner. Basic Books, New York (1983)
33. Gibson, J.J.: The ecological approach to visual perception. Houghton Mifflin, Boston (1979)
34. Rasmussen, J., Vicente, K.J.: Coping with human errors through system design: implications for ecological interface design. International Journal of Man-Machine Studies 31, 517–534 (1989)
35. Engeström, Y.: Learning by Expanding (1987)
36. Scaife, M., Rogers, Y.: External cognition: how do graphical representations work? International Journal of Human-Computer Studies 45, 185–213 (1996)
37. Hutchins, E.: Cognition in the Wild. MIT Press, Cambridge (1995)
38. Pirolli, P., Card, S.K.: Information foraging. Psychological Review 106, 643–675 (1999)
39. Brown, J.S.: Research That Reinvents the Corporation. Harvard Business Review 68, 102 (1991)
40. Fischer, G., Girgensohn, A.: End-user modifiability in design environments. In: Conference End-User Modifiability in Design Environments, pp. 183–192. ACM, New York (Year)
41. Fischer, G.: Seeding, Evolutionary Growth and Reseeding: Constructing. Capturing and Evolving Knowledge in Domain-Oriented Design Environments Automated Software Engineering 5, 447–464 (1998)
42. Fischer, G., Lemke, A.C., Mastaglio, T., Morch, A.I.: Using critics to empower users. In: Conference Using Critics to Empower Users, pp. 337–347. ACM, New York (Year)
43. McCarthy, J.F., Boyd, D.M.: Digital backchannels in shared physical spaces: experiences at an academic conference. In: CHI 2005 Extended Abstracts on Human Factors in Computing Systems, pp. 1641–1644. ACM, Portland (2005)
44. Hartmann, B., Doorley, S., Klemmer, S.: Hacking, Mashing, Gluing: A Study of Opportunistic Design and Development. Stanford HCI Group (2006)
45. Norman, D.: The Invisible Computer. MIT Press, Cambridge (1998)
46. Buxton, B.: Less Is More (More or Less). In: Denning, P.J. (ed.) The Invisible Future - the Seamless Integration of Technology in Everyday Life, pp. 145–179. McGraw-Hill, New York (2002)

An Ontology-Based Approach to Product Customization

Carmelo Ardito[1], Barbara Rita Barricelli[2], Paolo Buono[1], Maria Francesca Costabile[1], Rosa Lanzilotti[1], Antonio Piccinno[1], and Stefano Valtolina[2]

[1] Università degli Studi di Bari, Dipartimento di Informatica, via Orabona 4, 70125 Bari, Italy
{ardito,buono,costabile,lanzilotti,piccinno}@di.uniba.it
[2] Università degli Studi di Milano, Dipartimento di Informatica e Comunicazione, via Comelico 39, 20135 Milano, Italy
{barricelli,valtolin}@dico.unimi.it

Abstract. Mass customization refers to the increase in variety and customization of the manufactured products and services. It is now economically feasible thanks to the availability of computer-aided manufacturing systems, which allow people to customize standard products, and to Internet, through which many online retailers now operate, thus eliminating the constraints of physical shelf space and other bottlenecks of distribution that, in past years, prevented the production of niche products because of their high production costs. To permit mass customization, several software-based product configurators are available on the Web: they guide people in adapting a product to their needs and desires. A drawback of such configurators is the limited range of changes permitted. We present in this paper a system that gives people more freedom in creating products that best fit their desires, thanks to the use of an ontology, which models the possible product compositions that users can perform. The proposed solution is shown through a case study, which refers to furniture production.

Keywords: product customization, long tail, end-user development, ontology, knowledge management.

1 Introduction

Mass customization is the new frontier in business competition for both manufacturing and service industries. It permits an increase in variety and customization of the manufactured products and services, avoiding cost increase. This is made possible by the use of computer-aided manufacturing systems, which combine the flexibility of individual customization with the low unit costs of mass production processes, i.e. the production of large amounts of standardized products [1].

Mass customization is defined as the method for "effectively postponing the task of differentiating a product for a specific customer until the latest possible point in the supply network." [2]. Different types of mass customization have been proposed in literature [3]. One of these types is *adaptive customization*: it means that companies produce standardized products, which customers can modify by themselves to produce a version customized to their needs and desires [4], [5].

In order to allow adaptive customization, several software-based product configurators, also known as mass customization toolkit, design kit, or toolkit for user

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 92–106, 2011.

innovation and design, are now available on Internet: they guide users (i.e. customers) to add or to modify features of a product in order to make it more suitable to their needs and desires [6]. Examples of product configurators are provided by IKEA [7] and Nike [8]. By using the IKEA configurator, the customer can select a product from the catalogue, e.g. a wardrobe, and change some of its features, like the type of wood, the color, and the size. A limitation of such configurators is that the changes customers can perform are constrained within a limited range of possibilities [6], [9]. Referring again to the IKEA example, the user cannot add a third drawer to a wardrobe if it is designed with only two drawers.

Some people may feel this as a strong limitation to their creativity and needs. The approach we present in this paper aims at providing users with software environments in which they will have more freedom in creating products that best fit their desires. The motivation came from a company working in the Puglia region, Maiellaro s.r.l., which produces classic style furniture. This type of furniture is usually very expensive, thus the company would lose a lot of money if it remains unsold. To cope with this problem, their business process is very different than traditional furniture producers. As it will be described in more detail in Section 4, the company produces only pieces of furniture, which are ordered by customers, who look at the company catalogues and provide a sketch of each piece of furniture they want, which may be composed by parts chosen from different items in the catalogues, and assembled together.

By considering the case study of Maiellaro company, we describe in this paper a system, which allows customers to create the wanted furniture. According to the SSW methodology [10], [11], the system is composed by a network of software environments, each personalized to culture, skills and needs of the stakeholders involved in the design. Customers have much more freedom in designing their furniture, thanks to the use of an ontology that models the possible composition of different parts in a whole piece. The proposed solution can be applied to different types of products. It goes beyond the current mass customization approaches, like those implemented by product configurators, keeping low production costs and, at the same time, supporting creativity of customers and increasing their satisfaction in getting a product much closer to their needs and desires.

The paper is organized as it follows. Section 2 summarizes current trends of Mass Customization and the Long Tail model. Section 3 discusses current product configurators. Section 4 presents the Maiellaro case study and Section 5 the ontology-based approach. In Section 6, the developed system prototype is illustrated. Finally, Section 7 reports conclusions and future work.

2 Mass Customization

Our economy is increasingly shifting from a focus on a limited number of mainstream products and markets, going toward a huge number of niches. As the costs of production and distribution fall, there is less need to lump products and consumers into one-size-fits-all containers. Mass customization is the new frontier for both manufacturing and service industries; it is now possible since computer-aided manufacturing systems permit people to customize standard products, thus keeping the low production costs of mass productions [5], [6].

The demand for products not available in traditional bricks and mortar stores is potentially as big as for those that are. Without the constraints of physical shelf space and other bottlenecks of distribution, narrowly-target goods and services can be as economically attractive as mainstream fare.

Researchers refer to the Long Tail as the right part of the chart represented in Fig. 1, which shows a standard demand curve of any industry [12]. The horizontal axis represents products that can be manufactured by a certain industry (on the left the most common products, on the right niches products); the vertical axis represents sales frequency, dependent on the product popularity of each product (on the left most common products, on the right niche products). Mainstream products ("hits"), which have dominated our markets for most of the last century, are in the left higher part of the curve (Head). The right lower, but longer part (Long Tail), refers to niches of products, and indicates where the new growth is coming from, now and in the future.

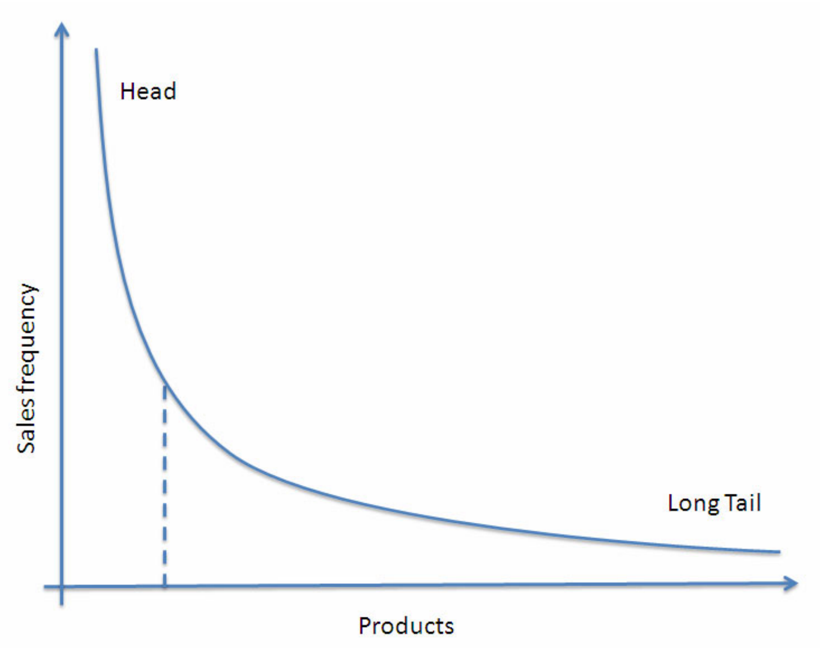


Fig. 1. The Long Tail model (adapted from [12])

According to traditional retail economics, stores only stocks the likely hits, because shelf space is expensive. In recent years, many online retailers, like Amazon [13], appeared on the market: they can stock virtually everything, so the number of niche products that are now available is larger than the hits by several orders of magnitude. These millions of niches are the Long Tail in Fig. 1; they have been largely neglected so far due to economic reasons. The variety of world population pushes towards niches because they satisfy narrow interests better. Today the Web has released the constraints of physical storage space, making possible to offer consumers many more choices.

In [12], an analysis of the sales data and trends in the digital entertainment market has shown that its economy is going to be radically different from today's mass market. While the 20th-century entertainment industry focused on hits, the 21st will focus on niches. It is even claimed that many of our assumptions about popular taste are artifacts of poor supply-and-demand matching, which is the market response to an inefficient distribution.

It is widely acknowledged that the new trend addressing the Long Tail is not limited to the entertainment market and will affect all manufacturing industries. An example referring to furniture manufacturing is reported in this paper.

3 Product Configurators

In the last years, companies have been taking into account opportunities that the long tail can give. In order to provide individual customers with unique products, mass customization strategies and tools have been developed.

Among the tools available on the market, product configurators are widely used and offer customers the possibility of adapting, to some extent, a product to their needs and desires by using a direct and visual version of the configured product. The aim is to let users to personalize the product [1], [6], [9]. A configuration is usually obtained in several steps because there may be several aspects of the product that can be configured, e.g. color, material, writings, etc. The product configurator is therefore a Wizard where every configurable aspect of the product is handled in a single step.

A product configurator is a highly visual interactive environment where users configure the product by direct manipulation. Every time users make a change, they immediately see the results. Users can “play” with products and “see” all available options. Finally, they obtain a view of the product they may want to order.

We analyzed several product configurators available on the Web. An example is NIKEiD, the shoe customization tool of Nike. Once a shoe model has been selected by the user (see Fig. 2), the system shows the shoe of which the user can visually identify all its components by moving the mouse pointer over and change each one by clicking on it and choosing among a proposed set of elements (shown in the upper-left part of Fig 2). In this way, the user can personalize the shoes in a number of steps, by changing materials and colors, adding personal ID, choosing among wide and narrow sizes, etc.

The user can also rotate the image, choose among predefined view (e.g., front or side view) and zoom it to highlight details. S/he is always driven throughout the personalization process by the configurator. Feedbacks about the status of the personalization are provided through the function “What’s left” that proposes the steps not yet performed. The configuration process ends after all the required steps (twelve in the example of Fig. 2) are performed. After this, the final customized shoe is presented together with the calculated price. The user can save, share (through e-mail or direct link) or order the model. NIKEiD is very easy to be used, it is Web based and it only requires the installation of the Flash player for visualizing the tool.

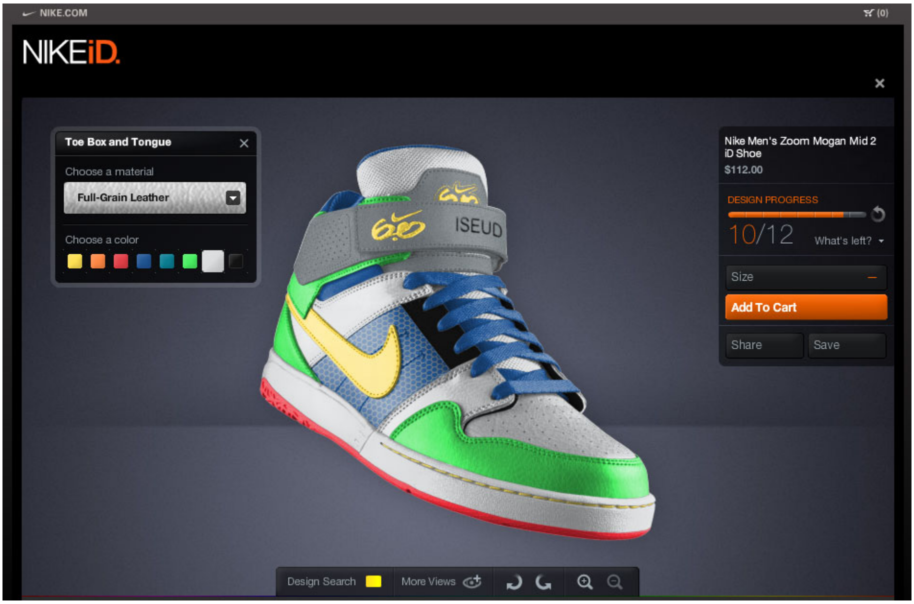


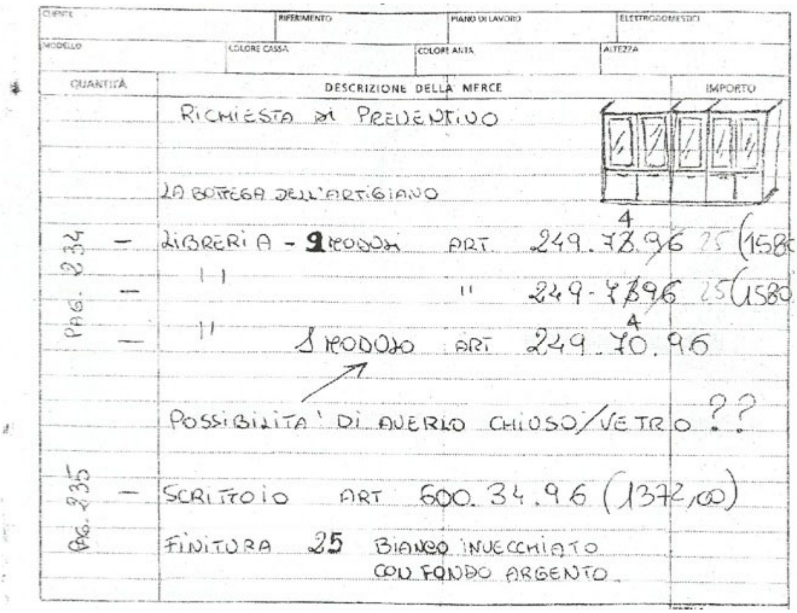
Fig. 2. The product configurator NIKEiD from Nike

A limitation of all product configurators is that only predefined changes can be performed by end users, i.e. only those ones that can be performed without changing the manufacturing system and without additional costs for the production chain. There are some cases, like the one reported in Section 4, in which people need to have much more freedom and should be allowed to design themselves the products of interest.

4 The Case Study

Maiellaro s.r.l. is a company producing classic style furniture. Since this furniture is very expensive, the risk of losing money if it remains unsold is very high. Thus, their business process is different than traditional furniture producers: they only produce those pieces that are ordered by a customer, who looks at the several catalogues the company provides and sends an order of a specific piece. In order to satisfy their customers as much as possible, the company wants to allow all freedom in designing a piece they want, which can be composed of parts chosen by different items in the catalogues and whose dimensions, type of wood and other characteristics are specified by the customers.

Fig. 3 shows a customer request for a personalized bookcase ("libreria" in Italian). The request contains a sketch of the bookcase and indicates references of the articles in Maiellaro's catalogues of which it is composed (e.g. art. 249.70.96) and further personalization requests (like the possibility to have it closed with a glass door). Current practice is that the customers send via fax to the company requests like this one. The company evaluates the feasibility of each request. If it can be satisfied, the price is negotiated through a communication exchange, via phone and/or fax, between company sale office and customer. Once the estimate price has been accepted, the company creates an internal document containing the description of the new piece of furniture and it production starts. This new piece will then be added to the Maiellaro's catalogues.



CLIENTE	RIFERIMENTO	PIANO DI LAVORO	ELETTRODOMESTICI
MODELLO	COLORE CASSA	COLORE ANTA	ALTEZZA

QUANTITÀ	DESCRIZIONE DELLA MERCE	IMPORTO
	RICHIESTA DI PREVENTIVO	
	LA BOTTEGA DELL'ARTIGIANO	
PAG. 234 –	LIBRERIA - 2 MODULI ART 249.78.96 4	25 (158
–	" " 249-78.96	25 (1580
–	" 1 MODULO ART 249.70.96 4	
	POSSIBILITA' DI AVERLO CHIUSO/VETRO	??
PAG. 235 –	SCRITTOIO ART 600.34.96 (1372,00)	
	FINITURA 25 BIANCO INVECCHIATO CON FONDO ARGENTO.	

Fig. 3. An example of request made to Maiellaro company

The Maiellaro's business process provides an interesting solution to the Long Tail problem, being able to satisfy customers desires, still keeping reduced production costs. This case is also very challenging from an ICT point of view [14]. In the next sections we show the approach we have adopted to design a system to be used by company customers to shape their own products.

5 Ontology-Based Approach

One of the problems of the case study presented in the previous section is that currently the information needed to customize a piece of furniture are scattered in different archives. These archives refer to the catalogue of Maiellaro company and to a set of other catalogues of Maiellaro's suppliers. These suppliers' catalogues are very diverse and heterogeneous because they refer to various crafts and arts (e.g. glass suppliers, wood artisans). Thus, it is really difficult to have an overall view on all existing information. Another problem is how to drive customers in selecting components of items in the catalogues and allow them to assemble such components in reasonable ways to create a new piece. We describe here our approach based on the use of ontologies to solve such problems.

First of all, a strategy to connect the objects that constitute the Maiellaro's catalogue with those that belong to the suppliers' catalogues is needed, so that customers can consider all of them as a unique catalogue. The integration of heterogeneous information sources implies the design of a data integration system aimed at dealing with data residing in several sources and at hiding to the user the source of the data s/he is accessing and its structure. The use of ontology for the explication of the knowledge is a possible approach to address such an integration problem. This solution is based on an important requirement for knowledge models: the ability to abstract from the different storage strategies of various catalogues.

Several research projects (e.g., [15], [16], [17]) have led to the definition of ontological models that allow one to describe a given application domain and then to retrieve the associated context information from distributed data sources [18]. Such researches have investigated the use of ontology schemas for modeling implicit and hidden knowledge in order to integrate different databases owned by different providers. For example, in a solution called "virtual approach" [17], [18] data residing at the sources are accessed during query execution, and are not replicated in the integrated system. This approach is adopted in order to use the knowledge base as a sort of semantic access point to the information that can be retrieved from different databases federated by means of an ontology schema. Archives owned by different suppliers can be mapped to each other and related independently from the craft or art to which they refer.

Looking at the literature, we found two cases of ontologies in the furniture context. The first case refers to the transformation of the Arts and Architecture Thesaurus into an ontology in order to capture background knowledge for antique western furniture [19]. In this case, each piece of furniture is considered as a whole and its parts are not described individually. The second case describes a knowledge management design method that aims at supporting designers in reuse of existing design cases [20]. The ontology doesn't describe a piece of furniture in its components but it is used to arrange furniture at "taxology level"; in other words, the ontology is used to help identifying pieces of furniture that can inspire the design of new ones.

For the Maiellaro's case study, an ontology has been defined, to support people customization of pieces of furniture. The customer, starting from furniture shown in the catalogues, is able to identify the combination of components for creating her/his wished piece of furniture. This combination is carried out in terms of substitution or application of components or decorations taken from other pieces of furniture or by means of changes about color, material or size of the components themselves. It should not be presented to the users a classification of all the possible combinations that they can request. On the contrary, customers should be able to combine pieces at their disposal, as they like at best.

The use of the ontology permits to describe the components of each piece of furniture and their properties; for each component, it is possible to specify colors, size, decorations, shapes, and materials. Moreover, the ontology provides all rules and constraints to be applied to assemble various components in order to generate all and only those pieces of furniture that are considered by the ontology. A large set of pieces of furniture is considered in this case study, organized in categories such as tables, desks, bookshelves, armchairs, sofas, wardrobes, consoles, etc. For example, a console is composed by a top, a set of drawers, a set of handles, and a set of legs, as shown in Fig. 4.

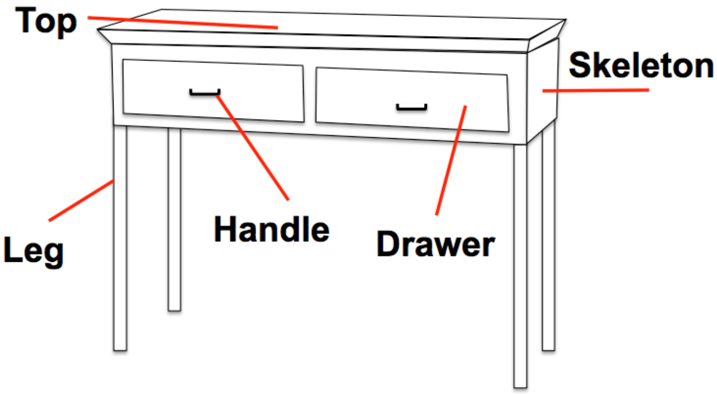


Fig. 4. A console and its components

Fig.5 illustrates a portion of the ontology related to the console concept. The classes are used to represent the relations existing among a set of components of a piece of furniture. Solid lines represent relations of subclass. For example: a console is a subclass of a piece of furniture ("mobile" in Italian); its components are: drawers ("cassetti"), legs ("gambe"), top ("piano"), handles ("maniglie") and skeleton ("scheletro"). Dashed lines represent relations of belonging; for example, a console is composed by drawers, legs, top and skeleton; a component is related to concepts such as decoration ("decorazione"), size ("dimensione"), color ("colore") and material ("materiale").

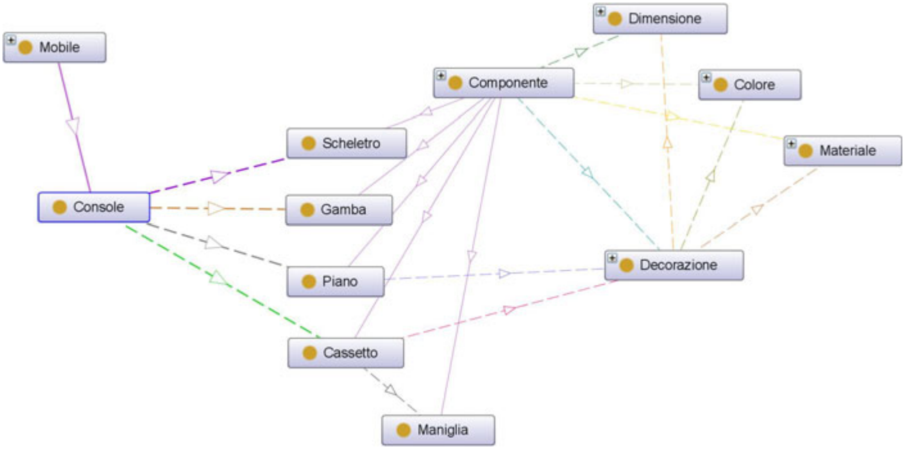


Fig. 5. A portion the ontology representing the console concept

From a technical point of view, the ontology uses a machine-readable format such as RDF. Therefore, class names and properties are encoded using RDF labels. Moreover, the mappings between the ontology and the relational schema of each database integrated in the system are encoded in RDF. In order to be able to represent the information mapping, the ontology has been extended by adding two classes: DB_Class_Mapping and DB_Property_Mapping. Because these classes do not model any domain concept, they have been placed outside the original Maiellaro's class hierarchy. The two classes are endowed with a set of properties, which refer the information related to the mapping between the ontology and each integrated DB. The information mapping inserted in the ontology permits to define transformation algorithms (implemented in JAVA), which translate a semantic query (expressed in SeRQL, an RDF Query Language) into SQL statements, one for each integrated database.

Going beyond standard digital retrieval operations, the system exploits the ontology expressing the concepts relevant for the domain and uses it to integrate the available data sources, providing a uniform point of access to all information. A semantic mediator allows the user to formulate queries in terms of the domain's concepts rather than entities defined in the databases' logical schemas; e.g., "retrieve all consoles having waxed wooden top" or "retrieve all furniture having decorated drawers". A query expressed by the user through a form-based interface is mapped onto a semantic query and from this query onto an SQL query. For example, a query to retrieve all consoles made of wood is first translated onto the semantic query:

```
SELECT Consoles
       FROM {Consoles} rdf:type {owl:E22.Man-Made_Object},
            owl:p45f.consists_of {Material};
                   {Material} rdf:type {owl:E57.Material};
                         {Material}rdfs:comment {MaterialName}
       WHERE label(MaterialName) like "wood“
            USING namespace owl = <"http://www.w3.org/2002/07/owl#">
```

and then onto in a set of SQL queries, e.g.:

```
SELECT ConsoleID
       FROM Material JOIN Consoles
              ON Material.Material = Consoles.MaterialID
       WHERE Material.Material=“wood"
```

The obtained SQL statements enable the access to the integrated databases by means of Sesame, an open source semantic Java framework.

The proposed ontology virtually unifies scattered catalogues of different suppliers; it instances a conceptual abstraction of the knowledge developed by experts. Using this system, customers can retrieve data from different catalogues and can combine components in order to create a new piece of furniture; the ontology determines which customer's compositions are acceptable.

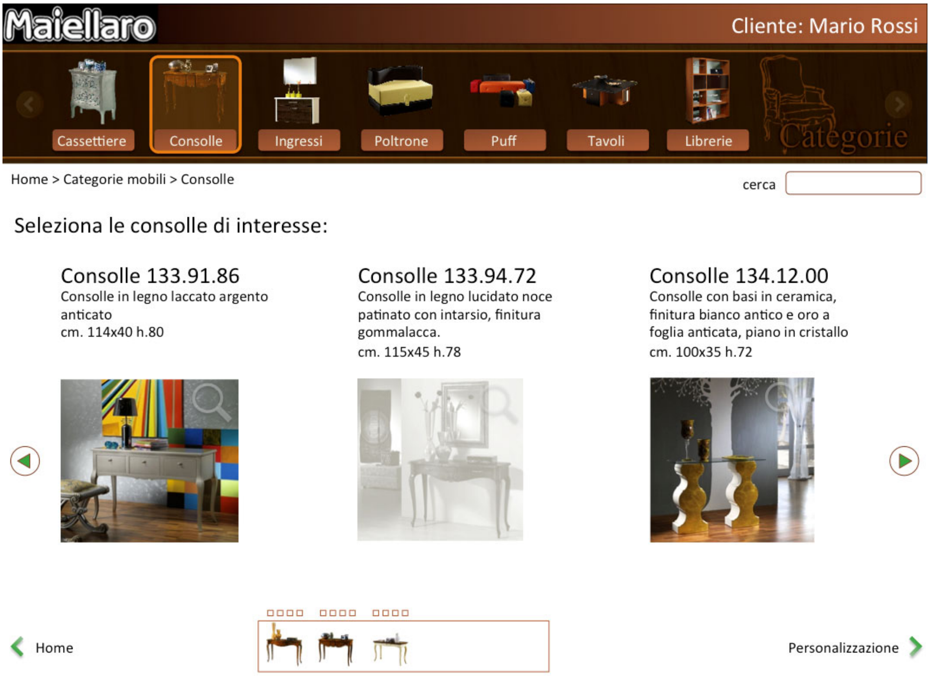
6 The Developed Prototype

In this section, we present the prototype of the system developed for the case study of Maiellaro company, in order to support the negotiation process between customers and the company finalized to order a piece of furniture and, more importantly, to allow customers to create the piece as they wish.

In designing the system prototype, we followed the SSW methodology, which foresees that all the involved stakeholders should actively participate to system design, being provided with suitable software environments, languages and tools to foster their personal and common reasoning about the development of systems that support end users' work [10], [11], [21]. These software environments are called *Software Shaping Workshops* (briefly, SSWs or workshops). The term *workshop* comes from the analogy with an artisan workshop (e.g., smith's workshop), i.e. the workroom where a person finds all and only those tools necessary to carry out her/his activities. Following this analogy, each community of experts participating in the design team, namely software engineers, HCI experts, domain experts, and end users, is provided with a workshop tailored to the experts' culture, through which they contribute to shape software artifacts, i.e. they access and modify the artifacts of interest, and exchange the results of their activities to converge to a common design. Thus, this approach fosters End-User Development and collaboration among all system stakeholders [22], [23]. To permit End-User Development, a new paradigm for the design of interactive systems is considered, called meta-design: professional developers act as meta-designers since, instead of developing the final interactive system, as in traditional design approach, they design software environments for the communities of stakeholders in the design team, who use such environments to collaborate in the whole life-cycle of an interactive system [11], [21], [24].

During the field study we carried out for requirements analysis at Maiellaro company, we found that the design team has to include, together with the professional developers with technical skills, namely software engineers and HCI experts, the following four stakeholders: (a) the managing director, who supervises the company business processes and, in particular, is in charge of the approval of the order of new pieces of furniture, designed by the customers, which will also be inserted in the catalogues; (b) sales office employees, who manage orders by customers in collaboration with the technical office; (c) customers, who order pieces of furniture by selecting items from the Maiellaro's catalogues and customizing them as they wish; (d) technical department employees, who manage technical aspects of new pieces of furniture and also have the responsibility of updating the ontology when new catalogues or new furniture are added.

In the following, we illustrate the workshop devoted to Maiellaro's customers. As described in Section 5, on the basis of the defined ontology, the customer is driven in performing customization activity. Let us suppose the customer Mario Rossi (male) wants to order a console of certain dimensions to fit his living room. He logs into the system on the Web and accesses the customer workshops. He is informed that he can order a piece of furniture by customizing items in the available catalogues. The customer then goes on to browse the catalogues, where products are organized by category, as shown in Fig. 6. He chooses an item of interest by clicking on its picture, and a thumbnail of that item appears in the box at the bottom of the screen.



Home > Categorie mobili > Consolle

Seleziona le consolle di interesse:

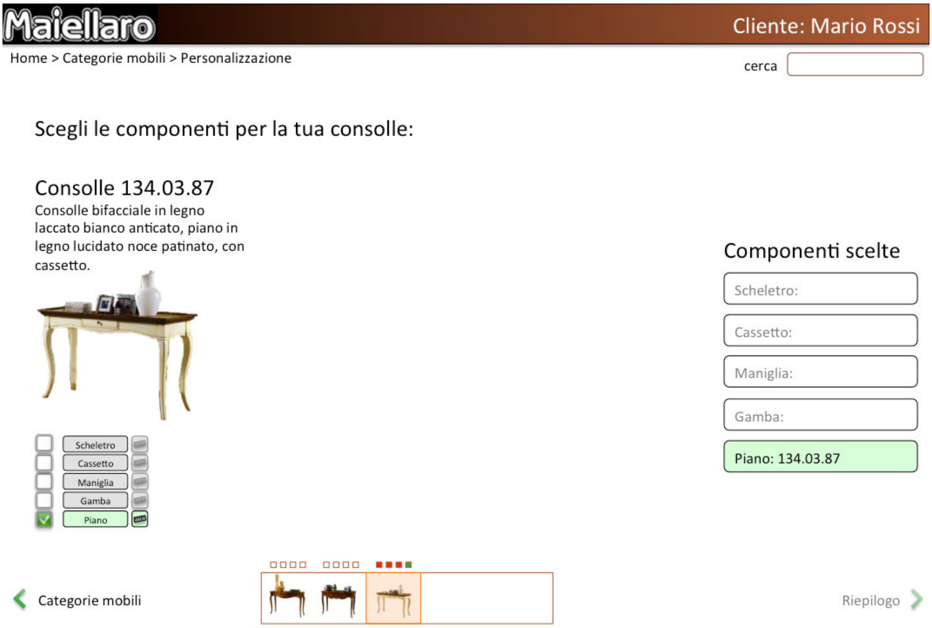
Consolle 133.91.86
Consolle in legno laccato argento anticato
cm. 114x40 h.80

Consolle 133.94.72
Consolle in legno lucidato noce patinato con intarsio, finitura gommalacca.
cm. 115x45 h.78

Consolle 134.12.00
Consolle con basi in ceramica, finitura bianco antico e oro a foglia anticata, piano in cristallo
cm. 100x35 h.72

Fig. 6. The customer Mario Rossi browses the catalogue and chooses three consoles of interest

When the customer has selected all items of interest, he goes on in creating the piece he wishes. This piece can be either a specific item he found in the catalogue, and for which the customer wants to modify only certain features, such as type of wood, size, etc., or it can be the result of a more sophisticated design process, i.e. the composition of parts taken from different items. For example, he would desire a console with parts taken from the selected items in the box at the bottom. He goes on with his customization process by clicking on the link at the bottom right of Fig. 6 "Personalizza"; a new screen appears where he can indicate the component of interest in each of the console previously selected. By clicking on the third thumbnail at the bottom, the picture of that console appears (console number 134.03.87), as shown in Fig.7. On the basis of the ontology (see Fig. 5), the system shows that, for console, the components that can be selected are: top ("piano"), leg ("gamba"), handler ("maniglia"), drawer ("cassetto"), skeleton ("scheletro"). The customer selects a component by clicking on the checkbox at left. In Fig. 7, Mario Rossi has selected the top of console n. 134.03.87. For this component, he can also specify material, color, size, etc. by clicking on the button at right of the label "piano". A pop-up window appears at the center of the screen (it is not shown in Fig. 7) and the customer specifies the values he wants. At the right part of Fig. 7, a summary of the selected components and the indication of the console from which each specific component comes is shown. In the example, only the top from the console n. 134.03.87 has been selected. The customers can go on selecting the remaining components from the same console, or he can click on another item in the box at the bottom of the screen, so that it will appear at the main area of the screen and the customer will select other components from it. The customer can also go back, clicking on “Categorie mobile” link, and choose other consoles, which will be added in the box of selected items.



Maiellaro Cliente: Mario Rossi

Home > Categorie mobili > Personalizzazione

cerca

Scegli le componenti per la tua consolle:

Consolle 134.03.87

Consolle bifacciale in legno laccato bianco anticato, piano in legno lucidato noce patinato, con cassetto.

- ☐ Scheletro
- ☐ Cassetto
- ☐ Maniglia
- ☐ Gamba
- ☑ Piano

Componenti scelte

Scheletro:

Cassetto:

Maniglia:

Gamba:

Piano: 134.03.87

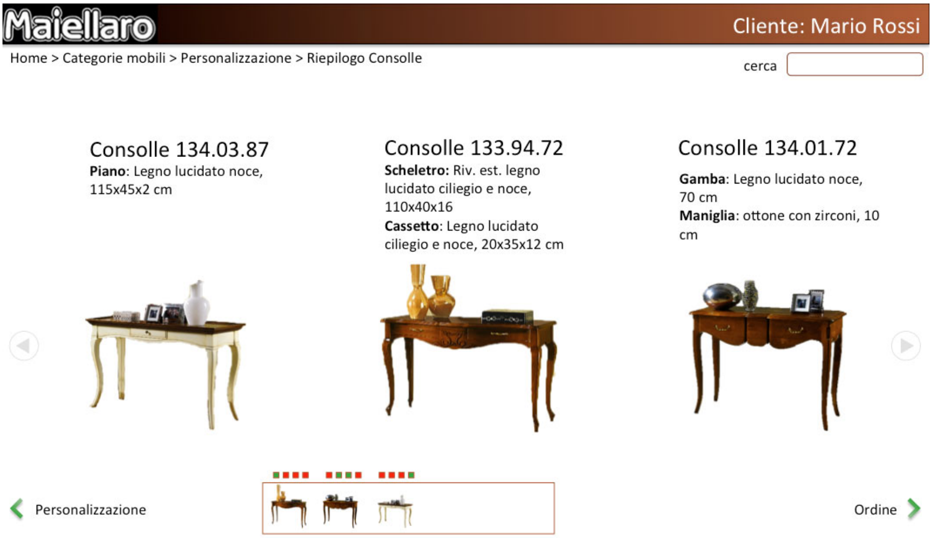
< Categorie mobili

Riepilogo >

Fig. 7. The customer is composing the console he wants to order and has chosen the top from the console number 134.03.87

At any moment, the customer can visualize his overall composition by clicking on the link “Riepilogo” at the screen bottom right and the screen in Fig 8 appears. It shows the catalogue items from which components that contribute to create the console the customers wants come from. Moreover, for each component the features specified by the customer, e.g., material, size, etc., are indicated. Once the user has completed the console s/he wants, a click on the link “Ordine” at the screen bottom right in Fig. 8 sends the order to the sales office, which checks the design created by the customer. The customer can then better see the overall design of the console s/he specified. Once sales office and customer agree on the price, the official order is delivered and the production of the new console starts.

As we said, other system stakeholders are the technical office employees, who are in charge of adding new catalogues provided by different suppliers in order to map them on the ontology, independently from their location and typology. To this aim, they have to handle the information retrieved from different databases as instances of the ontology classes and to establish a correspondence between the relations defined in the database schema and the classes in the ontology. The result of this process, iterated for each catalogue, is a semantic network able to translate the DB schemas onto concepts and relations of the ontology. The information defining the mappings is used by the system to generate SQL code for querying each DB schema. In this way, the query can be expressed independently from the query languages of the underlying database. Currently, this activity is carried out by using Protégé [25]. We are developing a new software environment to support the mapping process through a wizard that drives the user along the appropriate steps.


Maiellaro
Cliente: Mario Rossi
Home > Categorie mobili > Personalizzazione > Riepilogo Consolle
cerca
Personalizzazione
Ordine

Consolle 134.03.87
Piano: Legno lucidato noce, 115x45x2 cm

Consolle 133.94.72
Scheletro: Riv. est. legno lucidato ciliegio e noce, 110x40x16
Cassetto: Legno lucidato ciliegio e noce, 20x35x12 cm

Consolle 134.01.72
Gamba: Legno lucidato noce, 70 cm
Maniglia: ottone con zirconi, 10 cm

Fig. 8. Summary of console components chosen by customer Mario Rossi

7 Conclusion and Future Work

This paper has proposed a new approach to product customization, in order to give people more freedom in creating products that best fit their needs and desires. The motivation came from the Maiellaro s.r.l. company, which produces classic style furniture. Their business process is very different than traditional furniture producers, since they focus on niches of products rather than on mass production. In order to satisfy its customers as much as possible and to avoid producing furniture that might remain unsold, the company creates only pieces of furniture which are ordered by customers, who look at the company catalogues and design each piece of furniture they want, which may be composed by parts chosen from different items in the catalogues, and assembled together.

The system described in this paper for the Maiellaro case study allows customers much more freedom in customizing products than current product configurators, thanks to the use of an ontology that models the possible compositions of different parts in a whole piece, thus driving customers' design activities and ensuring that they may only perform acceptable modifications. A prototype of the system is described, which shows how customers can express their requests in order to create the furniture piece they wish and how they can, together with the company, finalize its order. This prototype has been recently developed and only formative evaluation has been performed so far, primarily with inspection methods and user testing through a thinking aloud method, involving four people. More accurate studies will be performed in the near future.

The proposed solution can be applied to different types of products, e.g., assembled computers, shoes, etc. It exploits the definition of a comprehensive knowledge base able to integrate heterogeneous data-sources and to contextualize their information with the active participation of domain experts.

A new possible research direction could address the study of an access control model based on the notion of group-based data sharing to apply to the ontology. Another research direction could be related to the use of the ontology for integrating data sources other than relational databases. In particular, a relevant direction is to use the ontology in order to support a semantic orchestration of web-services. In this scenario, the data sources are accessed using specific web services and the Maiellario's system is used for coordinating them according to the conceptual ontology.

Acknowledgments. This work is partially supported by Regione Puglia under grant "Tecnologie End-User Development per la personalizzazione di mobili classici italiani", POR Puglia 2007- 2013 – Asse I – Linea di Intervento 1.1 – Azione 1.1.2. We thank the Maiellaro company for the valuable collaboration. The work of Barbara Rita Barricelli and Stefano Valtolina is supported by the Initial Training Network "Marie Curie Actions", funded by the FP7 People Programme (reference: PITN-GA-2008-215446-DESIRE) entitled "DESIRE: Creative Design for Innovation in Science and Technology". We also thank the former student Antonio Caccavo, who participated to the development of the system prototype.

References

1. Simpson, T.W.: Product Platform Design and Customization: Status and Promise. Artif. Intell. Eng. Des. Anal. Manuf. 18(1), 3–20 (2004)
2. Chase, R.B., Jacobs, F.R., Aquilano, N.J.: Operations Management for Competitive Advantage, 11th edn. McGraw-Hill/Irwin, New York (2005)
3. Pine II, B.J.: Mass Customization: The New Frontier in Business Competition. Harvard Business School Press, Boston (1993)
4. Zhang, X., Yang, Y., Liu, S., Liu, F.: Realization of a Development Platform for Web-Based Product Customization Systems. Int. J. Comput. Integr. Manuf. 20(2-3), 254–264 (2007)
5. Zha, X.F., Sriram, R.D., Lu, W.F.: Evaluation and Selection in Product Design for Mass Customization: A Knowledge Decision Support Approach. Artif. Intell. Eng. Des. Anal. Manuf. 18(1), 87–109 (2004)
6. Franke, N., Schreier, M., Kaiser, U.: The "I Designed It Myself" Effect in Mass Customization. Management Science 56(1), 125–140 (2009)
7. IKEA: Ikea, `http://www.ikea.com/` (last access on December 26, 2010)
8. NIKEiD: Nikeid, `http://nikeid.nike.com/nikeid/` (last access on December 26, 2010)
9. Trentin, A., Perin, E., Forza, C.: Overcoming the Customization-Responsiveness Squeeze by Using Product Configurators: Beyond Anecdotal Evidence. Computers in Industry 62(3), 260–268 (2011)

10. Costabile, M.F., Fogli, D., Fresta, G., Mussio, P., Piccinno, A.: Building Environments for End-User Development and Tailoring. In: IEEE Symposium on Human Centric Computing Languages and Environments, pp. 31–38. IEEE Computer Society, Auckland (2003)
11. Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A.: Visual Interactive Systems for End-User Development: A Model-Based Design Methodology. IEEE T. Syst. Man Cy. A 37(6), 1029–1046 (2007)
12. Anderson, C.: The Long Tail: Why the Future of Business Is Selling Less of More. Hyperion, New York (2006)
13. Amazon: Amazon.Com, `http://www.amazon.com/` (last access on December 26, 2010)
14. Oh, Y., Gross, M.D., Ishizaki, S., Yi-Luen Do, E.: A Constraint-Based Furniture Design Critic. Research and Practice in Technology Enhanced Learning (RPTEL) 5(2), 97–122 (2010)
15. Thomas, J.C., Kellogg, W.A., Erickson, T.: The Knowledge Management Puzzle: Human and Social Factors in Knowledge Management. IBM Syst. J. 40(4), 863–884 (2001)
16. Wexler, M.N.: The Who, What and Why of Knowledge Mapping. Journal of Knowledge Management 5(3), 249–263 (2001)
17. Lenzerini, M.: Data Integration: A Theoretical Perspective. In: 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2002), pp. 233–246. ACM Press, Madison (2002)
18. Valtolina, S.: Design of Knowledge Driven Interfacesin Cultural Contexts. International Journal on Semantic Computing 5(3), 525–553 (2008)
19. Wielinga, B.J., Schreiber, A.T., Wielemaker, J., Sandberg, J.A.C.: From Thesaurus to Ontology. In: K-CAP 2001, pp. 194–201. ACM Press, Victoria (2001)
20. Junhua, W., Fenghu, W., Weihong, S.: Knowledge Ontology Based Management System of Furniture Design. In: International Forum on Computer Science-Technology and Applications (IFCSTA 2009), pp. 282–285. IEEE Computer Society, Chongqing (2009)
21. Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A.: End-User Development: The Software Shaping Workshop Approach. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development, vol. 9, pp. 183–205. Springer, Dordrecht (2006)
22. Lieberman, H., Paternò, F., Wulf, V. (eds.): End User Development, vol. 9. Springer, Dordrecht (2006)
23. Sutcliffe, A., Mehandjiev, N.: Introduction Special Issue: End-User Development. Communications of the ACM 47(9), 31–32 (2004)
24. Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A., Mehandjiev, N.: Meta-Design: A Manifesto for End-User Development. Communications of the ACM 47(9), 33–37 (2004)
25. Protégé user documentation: Protégé User Documentation, `http://protege.stanford.edu/doc/users.html` (last access on December 26, 2010)

End-User Development of e-Government Services through Meta-modeling

Daniela Fogli and Loredana Parasiliti Provenza

Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Brescia
Via Branze 38, 25123 Brescia, Italy
{fogli,loredana.parasiliti}@ing.unibs.it

Abstract. This paper describes an approach to the end-user development of online services for citizens of a government agency. With reference to a typical government-to-citizen service, the paper discusses how such services are currently structured and provided to citizens, and how their implementation can be transferred from software professionals to administrative personnel, who do not generally possess any programming expertise. The analysis of e-government services is carried out according to different perspectives pertaining to the citizen, the employee, the software engineer and the human-computer interaction expert. This analysis leads to define an abstract service model (a meta-model) and constitutes the first phase of the end-user development approach here proposed. The meta-model can then be used to design an environment for service creation suitable to the competencies and background of the target end-user developers. This design activity constitutes the second phase of the proposed approach.

Keywords: e-government, XML, end-user developer, meta-model.

1 Introduction

The term "e-government" is usually adopted to denote web-based services provided by agencies of local, state and federal governments. Such services can be supplied to several entities, from citizens and business enterprises to government agencies, thus being classified into Government-to-Citizen (G2C), Government-to-Business (G2B), and Government-to-Government (G2G) services [23].

The Department of Information Engineering of the University of Brescia is involved in an ongoing collaboration with an Italian government agency, the Brescia Municipality, for designing and developing G2C services. This collaboration has started in 2007 in the context of e-government web content creation and accessibility [9][10]. During this work we extended the web publishing system used by administrative employees with End-User Development (EUD) functionalities for generating accessible content.

Within this collaboration, a further and more ambitious need emerged during the meetings with software developers working at the Brescia Municipality: empower

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 107–122, 2011.

administrative employees to create e-government G2C services, such as front-office reservation, online tax payment, enrollment in schools, etc. Notice that, in this case, administrative employees should become *end-user developers* who create online services to be exploited by citizens, i.e. the actual end users. However, administrative employees, although expert in the government domain, are usually neither expert in information technologies nor motivated to learn them. Such domain experts usually possess specific competencies, for example they know the different kinds of data to be acquired and the procedures to be carried out for the payment of local taxes. At the same time, they feel at ease with software systems only when these systems adopt interaction styles consistent with the software applications they commonly use in their work and daily practices, e.g. browsers, spreadsheets or word processors.

These considerations led us to devise an EUD approach, which aims to allow administrative employees to act as unwitting service developers [6]. The approach explores the different perspectives of all the stakeholders involved in service creation and use, and capitalizes on the core concepts of these different perspectives to design an EUD environment suitable to administrative employees' profile. To this end, the approach consists of two main phases:

1) derive an abstraction of e-government services on the basis of the different descriptions of a service that the stakeholders involved in its implementation and use can provide. This activity leads to the definition of a *meta-model* of e-government services;

2) design and develop an interactive software application that supports domain experts (i.e. administrative employees) in creating an instance of the above meta-model, that is a *service model*. The same software application should also be able to interpret the service model and automatically generate the service itself.

The first phase is a typical software engineering task, which however considers the domain and Human-Computer Interaction (HCI) experts' perspectives to arrive at a richer meta-model. Whilst, the focus of the second phase is on designing the interaction experience offered by an EUD environment that could foster employees' participation in service creation, without asking them to acquire additional competencies in information technologies.

The paper is organized as follows. Section 2 analyses the characteristics of current e-government services, with reference to the real case of Brescia Municipality. Section 3 describes our two-phase approach to the development of an EUD environment for e-government service creation. Section 4 starts by discussing the profile of the different stakeholders in the considered domain and then presents the meta-model synthesizing the different stakeholders' points of view on e-government services. Section 5 illustrates the EUD environment allowing employees to become unwitting programmers of e-government services. Section 6 then compares our approach with literature work, and, finally, Section 7 concludes the paper.

2 e-Government Services

Electronic service delivery (ESD) usually refers to providing government services through the Internet or other electronic means [31]. With ESD, federal, state, province and local government agencies may interact with citizens and organizations (business and other government organizations) to satisfy their many and various needs. Particularly, ESD provides a way to deliver services (e.g. paying taxes or getting a driver's license) directly to citizens, without requiring them to go to a government office. The European Union (EU) has defined five different levels of services that can be provided through the Web by government agencies [3]:

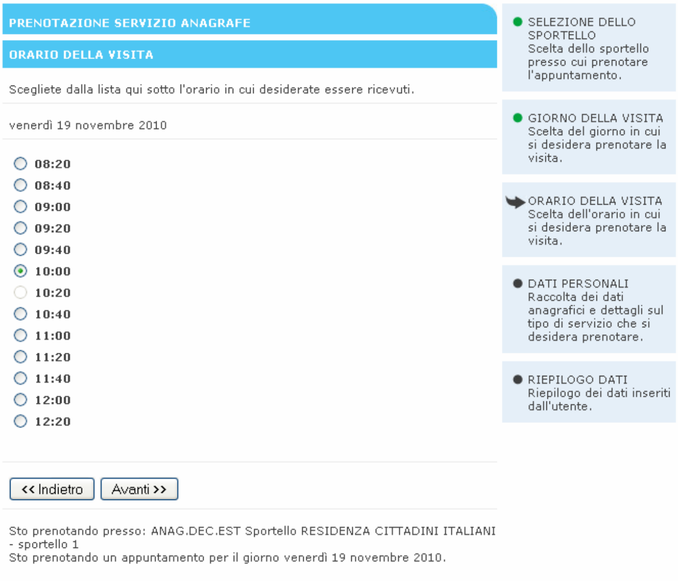
- *Level 1 – Information*: the website provides only information about the service and how it is supplied.
- *Level 2 – One-way interaction*: the website allows citizens to download the form to request the service; the filled form can then be sent to the government agency by means of traditional channels.
- *Level 3 – Two-way interaction*: the website allows citizens to start the service supply; for example, it is possible to fill in the service forms and submit them through the website.
- *Level 4 – Transaction*: the service is supplied completely on the website, possibly including its payment.
- *Level 5 – Personalization*: besides having the service completely supplied online, the citizen receives information (e.g. due dates, service outcome) on the basis of her/his profile.

The creation of services at levels 1 and 2 is concerned with the traditional activity of content creation in content management systems, and was already addressed in a previous collaboration with the Brescia Municipality [9][10].

In this paper, we focus our attention on the creation of e-government services belonging to levels 3, 4 and 5. These services have been further classified by domain experts and software developers working at Brescia municipality according to the following task-oriented characterization: i) front office reservation; ii) tax payment; iii) document request; iv) document submission; v) enrollment in courses or schools. Several services belonging to these five categories are made available to Brescia citizens through the municipality website (http://www.comune.brescia.it). They have been implemented according to a form-based metaphor, which reminds the approach citizens adopt in exploiting traditional government services. In the real world, citizens are usually given a sequence of paper forms to be filled in and submitted to a counter for receiving a utility. Similarly, in the virtual world, e-government services have been implemented as a set of web pages, organized as virtual forms, to be filled in and submitted to the agency when all data are provided and the necessary operations carried out.

For example, the service for booking appointments at the general registry office is implemented in the municipality website as a 5-step process including the following steps: 1) front-office selection; 2) date selection; 3) time selection; 4) input of personal data including appointment topics; 5) summary of inserted data and confirmation. In each step, citizens are provided with a form, which contains a limited number of widget types for data insertion (text fields, radio buttons, combo boxes and check boxes). As shown in Figure 1, in each form, a section placed on the right side of the screen visualizes the steps performed, the step under execution, and the remaining steps. More specifically, Figure 1 shows a portion of the web page during the booking of an appointment at the general registry office while the third step is under execution.

The form-based metaphor designed for e-government services has proved to be quite adequate to the heterogeneous population of potential users, due to its low cognitive burden and because it reminds the traditional interaction with paper-based forms. At present, the available online services are daily used by many citizens living in Brescia, even though alternative ways to access the same services are foreseen, e.g. phone calls or direct access to the competent office[1].



PRENOTAZIONE SERVIZIO ANAGRAFE

ORARIO DELLA VISITA

Scegliete dalla lista qui sotto l'orario in cui desiderate essere ricevuti.

venerdì 19 novembre 2010

○ 08:20
○ 08:40
○ 09:00
○ 09:20
○ 09:40
◉ 10:00
○ 10:20
○ 10:40
○ 11:00
○ 11:20
○ 11:40
○ 12:00
○ 12:20

Sto prenotando presso: ANAG.DEC.EST Sportello RESIDENZA CITTADINI ITALIANI - sportello 1
Sto prenotando un appuntamento per il giorno venerdì 19 novembre 2010.

● SELEZIONE DELLO SPORTELLO
Scelta dello sportello presso cui prenotare l'appuntamento.

● GIORNO DELLA VISITA
Scelta del giorno in cui si desidera prenotare la visita.

➡ ORARIO DELLA VISITA
Scelta dell'orario in cui si desidera prenotare la visita.

● DATI PERSONALI
Raccolta dei dati anagrafici e dettagli sul tipo di servizio che si desidera prenotare.

● RIEPILOGO DATI
Riepilogo dei dati inseriti dall'utente.

Fig. 1. Step 3 of the service for booking appointments at the general registry office

3 Toward End-User Development of G2C Services

The development of e-government services supplied to Brescia citizens is currently in the hands of the personnel of the Computer Science Department of the Brescia Municipality. Such activity, indeed, requires competences and programming skills that can be hardly found outside this department. Since G2C services are usually characterized by a recurring structure, their creation generally leads to implement very similar programs. In the past, software developers proceeded by replicating and adapting the code of another service, whilst nowadays they execute several configuration operations through an ad-hoc configuration system that carries out automatically almost all the coding activity. However, the core problem is that only administrative employees, as experts of government procedures, possess the know-how to structure and characterize a given service. To obtain the necessary input and configuration parameters for building the new service, several interactions between software developers

[1] For example, according to municipality studies, 70% of enrollments in schools and courses in 2009 were carried out through online services.

and administrative personnel usually take place, not rarely affected by misunderstandings and ambiguities. This work practice is unsatisfactory for both software developers and administrative employees: the former regard it as inefficient and ineffective, while the latter saw it as boring and time consuming. There is actually the need for a new perspective on service creation, which goes beyond the simple transfer of domain knowledge from employees to software developers.

To satisfy this need, the idea emerged from our collaboration with software developers at Brescia municipality is to make administrative personnel directly in charge of creating online services by using a suitable software tool. An important hurdle to this new work practice is however due to the fact that these personnel are not interested in contributing to online service creation if this implies learning new software tools and acquiring new practices or skills alien to their daily work and expertise. This state of affairs means that traditional end-user programming techniques foreseen in the end-user development of software artifacts, such as programming by example, component-based programming, visual programming, are not suitable to our end-user developers' profile, being, at present, too much computer-oriented. On the contrary, we need to provide administrative personnel with the 'right' EUD environment, in order to favor their progressive "change in mindset and culture" [8]. Such an environment should require a low cognitive burden and should support employees' best practices, being centered on the employees' idea of service creation in the paper-based world, rather than on the idea of 'creating a program' as the other EUD techniques actually are based on.

Coping with this problem requires an extended investigation, which starts from the analysis of existing e-government services, and arrives at understanding employees' preferences and computer skills, with the goal to develop a suitable EUD environment. The former is a bottom-up activity aimed at describing the 'essence' of an e-government service by abstracting from its software implementation, the interaction experience it offers and the mental model that administrative employees associate with it. The latter is a top-down activity aimed at designing an EUD system that allows end-user developers to create instances of the abstract service description without being aware of that. These instances are actually descriptions of programs implementing online services that can be automatically generated by the EUD system through a proper interpretation of descriptions themselves.

The following sections deepen these bottom-up and top-down phases.

4 From e-Government Service Analysis to a Service Meta-model

In the development of current web-based applications, including e-government services, a variety of aspects and issues must be taken into account: (i) the technologies involved; (ii) the interaction experience one would like to provide users with; (iii) the different usages recognized by users; (iv) the domain competencies necessary to develop useful and pertinent applications. All these aspects call for the analysis of the different stakeholders involved in software development, with the aim to overcome the limitations of the classical user-designer dichotomy. To this end, we analyze in the following the characteristics of stakeholders involved in e-government service creation and use, in order to derive an appropriate (meta-)model of services.

4.1 A Multi-faceted Description of e-Government Services

The stakeholders involved in our case study are: 1) the *citizens*, who need to exploit e-government services; 2) the *administrative employees*, who are required to actively participate – as experts of the government domain – in G2C service creation through a proper EUD environment; 3) the *HCI experts*, who are in charge of designing the interaction experience offered by such an EUD environment, suitable to the administrative employees; 4) the *software engineers,* who should finally develop such environment, satisfying the requirements established with the other stakeholders.

Since the profiles of HCI experts and software engineers are quite standard and well known, we prefer to dwell upon the characteristics of the other classes of stakeholders.

As to the citizens' profile, it refers to a high variety of people living in Brescia who may access the website of the municipality for service supply. Their age can vary in a wide range (from approximately 16 to 65 years old), as well as their education, cultural background and software technology knowledge. Most of them are often accustomed to interact with e-commerce web applications or social networks, whose interaction experience can therefore be exploited in the design of current online services. Anyway, given the high variety of the citizen profile and the importance of the goals that e-government services usually pursue, a careful attention should be paid on the usability of such services. They should guide citizens throughout the process until a successful task completion, without overwhelming or confusing them. Clear explanations should be provided, and the user should be able to move within the service at his own pace, but prevented to make irreparable errors or mistakes. The e-government services currently available on the Brescia Municipality website (as on many other Italian and world-wide government websites [21][25]) aim to meet and support the citizens' needs through the adoption of interaction design patterns like *wizard* and *fill-in-the-blanks* (or *form*) [29][30], which are suitable to guide users in using the online services.

As to administrative employees' profile, government agency employees are expert in different administration issues, and have different competencies, skills, and cultural background. Most of them are female, their age ranges approximately from 20 to 60 years old, and they do not usually hold a high education degree. They are usually acquainted with web browsers, word processors, spreadsheets and other similar office applications, but complain when they are charged by software systems with housekeeping activities. Additionally, as already said, they are not very motivated to participate in web content authoring that requires them to learn software technicalities.

Each stakeholder – citizen, employee, HCI expert, software engineer – has her/his personal view of e-government services, which can be described by a model that includes only the detail the stakeholder is interested in. The analysis of these different models, referred to the e-government service introduced in Section 2, has allowed us to define a service meta-model useful to design an EUD environment for administrative employees.

Figure 2 illustrates such a multi-faceted model of front-office reservation services.

A citizen usually describes an interactive system in natural language, possibly making reference to some screenshots of the system for a better explanation. Therefore, a possible description of the e-government service is the following: "*To obtain an appointment at the general registry office I should fill my data in these pages. For example, here* [screenshot with step 2] *I can select the date where I would like to be received. Then, I can select the button 'next' to reach this new page* [screenshot with step 3] *where I can find all the time slots available for booking appointments in the selected date. Therefore, I can click one of these choices and then go to the next page* […]".

An employee describes the e-government service using natural language as well, but s/he prefers focusing on the kind of data to be acquired and the procedures to be carried out to supply the service. An example description is the following: "*I must acquire some personal data of the citizen: name, surname, place and date of birth, tax code. Then, these data must be validated checking the tax code against the other data* […]".

An HCI expert, instead, analyzes an interactive system in terms of the interaction experience offered to its users. As HCI experts, we have described the given service according to Tidwell's interaction design patterns [29][30], thus focusing on the service metaphor and interaction style offered to citizens. We observed that, beyond the wizard and fill-in-the-blanks patterns, the developers adopted, among others, the patterns *center stage*, *go back one step*, *disable irrelevant things*, *choice from a small set*, *good default*, *interaction history*.

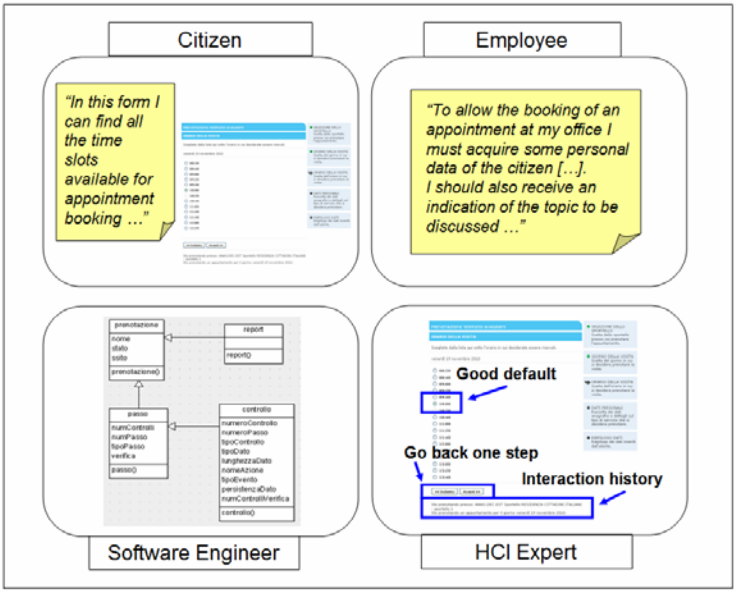


Fig. 2. The multi-faceted description of e-government services

Finally, a software engineer usually adopts a formal or semi-formal specification, such as an UML diagram. In our case, the software developer of the front-office reservation service cited above provided us with a high-level class diagram that describes the main classes of the software program implementing the service.

4.2 A Meta-model for e-Government Services

From the different stakeholders' models we arrived at defining an abstract model of e-government services, which actually plays the role of a meta-model. This meta-model has been described by an XML schema. The most significant part of this schema is illustrated in Figure 3 by means of a graphical representation. As one can notice in this figure, a `service` is structured as a sequence of one or more `step` elements, each one corresponding to a form through which (i) employees usually ask for the information needed and (ii) citizens provide the information required.

This conception of a service as a sequence of steps is, indeed, in accordance with the conceptual separation of the different kinds of data to be acquired (by the employee) and to be supplied (by the citizen). It is also compliant with a simple guided interaction, satisfying the wizard pattern, and with an implementation that represents both services and step as software objects.

A `step` is in turn a sequence of `requests` to the citizen on behalf of the employee, and in some particular case it may call a `substep`, which includes a sequence of `requests` as well. A `request` is always characterized by a `goal`: it can be a request for information or a request for data confirmation. A `request` further includes some `instructions` to guide the citizen while answering the request itself and is composed of one or more single items (`item` elements) and/or one or more groups of items (`item_group` elements), which represent the input data to be provided by the citizen to obtain the service. An `item` has some mandatory and optional attributes: for example, `label` must be assigned the value of the data label to be visualized to citizens, while `type_item` may potentially characterize the data to be acquired as a `name, surname, date, taxcode`, etc., in order to identify the procedure for its validation. For the software engineer a `request` is an object referring to other objects `(item` or `item_group)`. Whilst the HCI expert conceives a `request` as a set of text fields, combo boxes, check boxes, radio buttons, namely the virtual entities [5] to be used for visualizing and acquiring a single `item` or multiple items related each others (`item_group`).

Each `item` in turn may include zero or more `constraints` for the generation of data to be visualized, such as for example all dates and times available for booking appointments. In other cases, an `item` can be specified as a set of `options` to be shown to the user for selection, such as for example the topic to be discussed during the appointment.

In summary, the meta-model describes the main concepts of a service, which can be interpreted and specialized by the different stakeholders according to their point of view (usually as software objects, as interaction styles and widgets, or as different kinds of data to be received/supplied).

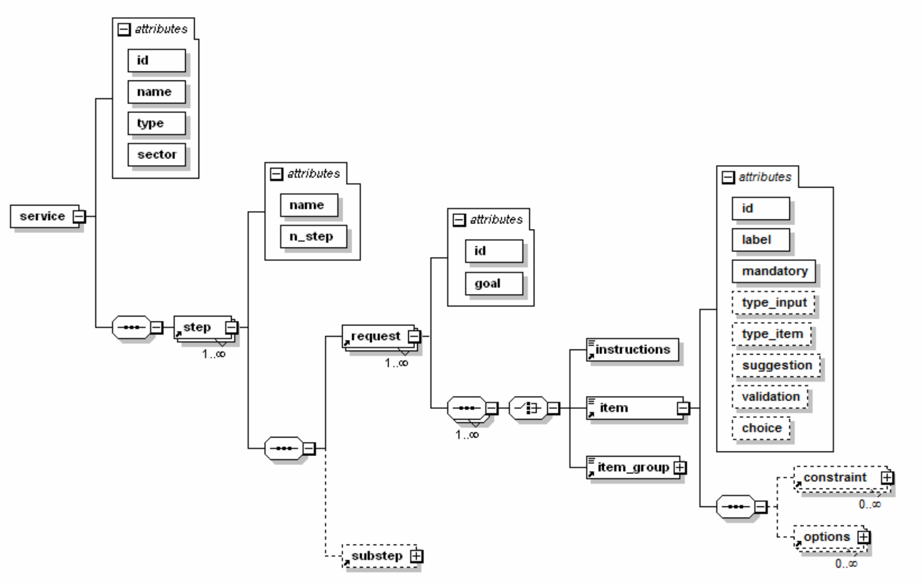


Fig. 3. Graphical representation of the XML Schema generated through Altova XMLSpy®

These different points of view have been taken into account in the design of the EUD environment for e-government service creation, which is described in the next section.

5 An EUD Environment for Service Creation

Starting from the abstract service specification described in the previous section, we have designed an EUD environment that empowers administrative employees to unwittingly implement e-government reservation services. Such environment drives end-user developers in generating the XML document (i.e. the service model) as an instantiation of the service meta-model. This environment does not force users to write any XML code, neither to know the underlying meta-model, but allows them to follow their usual way of reasoning and operating when a new G2C service has to be supplied. The EUD environment is designed to properly interpret the service model in order to generate the actual online service to be used by citizens.

More concretely, we have designed an EUD environment to guide employees in defining `steps`, `requests`, `items` etc., which characterize the `service` to be supplied. In designing the user interface, and particularly, the metaphor and the interaction style of the environment, we have paid attention on the employees' work practice when supplying a service. These stakeholders usually have a set of drafted forms (paper or electronic-based) they adapt and combine to provide the citizen with the forms to be filled in with the required information. Therefore, we can observe that the activity performed by employees requires again managing and interacting with sets of forms to be composed and/or adapted according to the service requirements. To this end, the environment for e-government service creation has been conceived as a service itself (namely, a meta-service) providing employees with the same interaction experience offered by online services to citizens, including editing content, selecting content from available choices, pasting content from other sources.

Figure 4 shows the EUD environment[2] developed for the creation of front-office reservation services[3], while an employee is creating the service “Residenza Cittadini Comunitari” (residence for EU citizens) for the “Settore Anagrafe” (registry office) of the municipality. According to employees’ work practices, the central part of the EUD environment (part 1 in Figure 4(a)) is always structured as a form to be filled in by the employee who is creating a step of the service. The employee can decide first which step s/he would like to create by selecting one of the buttons at the top of the page (part 2 in Figure 4(a)). In Figure 4(a), the employee is creating the step “Giorni di erogazione” (date selection). S/he has selected the corresponding button on the top of the page and s/he is filling out the central form with the constraints on the possible dates to be chosen for appointments. Here, the end-user developer establishes the start and final dates of the considered period, the number of weeks to be visualized, the days of the week when the office is open (Monday and Tuesday), and holidays. The help section on the left (part 3 in Figure 4(a)) provides the employee with indications on how to interact with the environment. When the end-user developer is satisfied by her/his definition, s/he can select the button “OK” and a new step of the service under construction will be added as a selectable widget on the right-side of the screen (part 4 in Figure 4(a)).

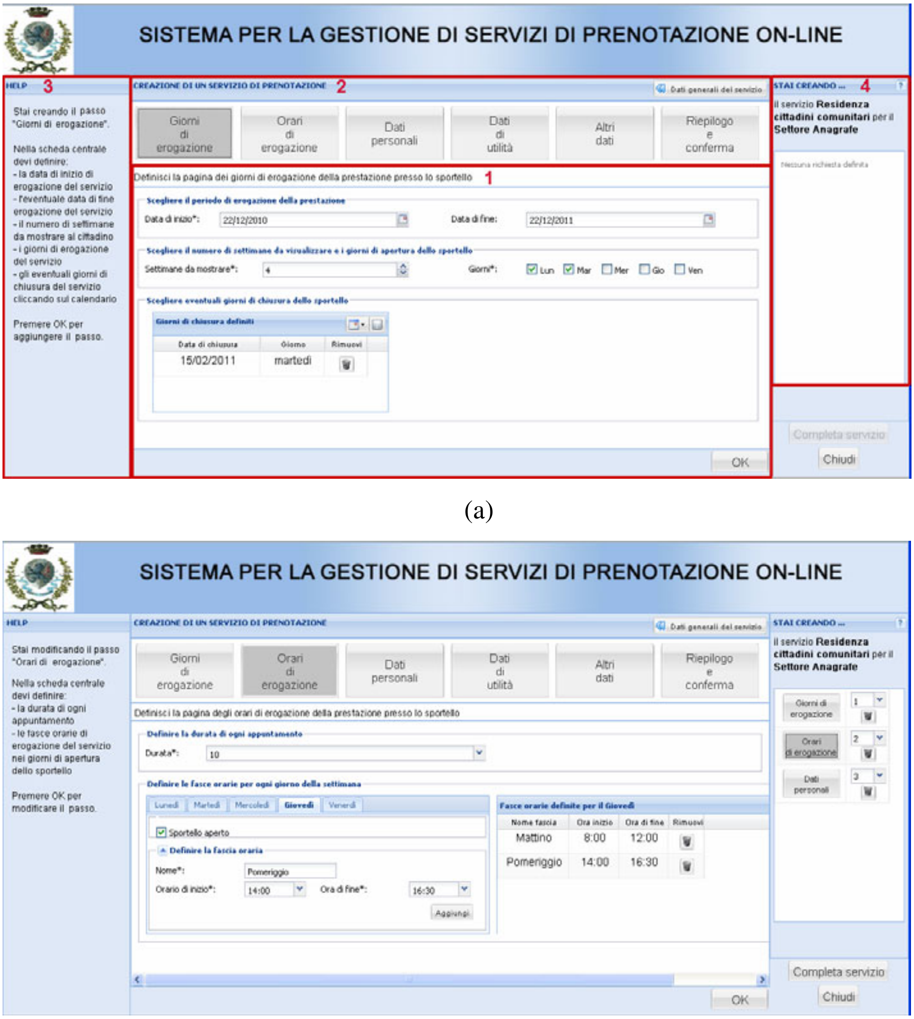
Figure 4(b) shows the EUD environment where the employee has already created the steps: “Giorni di erogazione”, “Orari di erogazione” (time selection) and “Dati personali” (input of personal data), as displayed in the right-hand side of the screen. In this part of the interface, the employee can decide to modify the step order by changing the sequence number of steps through the related combo boxes; s/he can also delete a step by clicking the corresponding wastebasket, or modify a step by selecting the corresponding button, thus recalling the inserted information in the central form. In Figure 4(b) the employee has selected the second step to change the inserted constraints on the office hours for appointments. Here, the employee can modify the inserted appointment duration and the time bands related to the open week days.

The information specified through the form in Figure 4(b) will be codified in the XML document and used by the environment to generate the step for time selection like the one shown in Figure 1.

The EUD environment also assists the employee entering consistent information while creating an e-government service. The information set in forms for dates and times are related each other: the end-user developer can operate independently on them, but the system keeps them always consistent; for example, the user can define the time band for a week day not selected in the form for defining available dates, and the information about the open week days is automatically updated in that form.

[2] The system is in Italian, in accordance with the profile of our end-user developers.

[3] In this current version the EUD environment allows the creation of e-government reservation services only, but it can be easily extended to the creation of other services.



Stai creando il passo "Giorni di erogazione".

Nella scheda centrale devi definire:
- la data di inizio di erogazione del servizio
- l'eventuale data di fine erogazione del servizio
- il numero di settimane da mostrare al cittadino
- i giorni di erogazione del servizio
- gli eventuali giorni di chiusura del servizio cliccando sul calendario

Premere OK per aggiungere il passo.

Definisci la pagina dei giorni di erogazione della prestazione presso lo sportello

Scegliere il periodo di erogazione della prestazione

Data di inizio*: 22/12/2010 Data di fine: 22/12/2011

Scegliere il numero di settimane da visualizzare e i giorni di apertura dello sportello

Settimane da mostrare*: 4 Giorni*: ☑ Lun ☑ Mar □ Mer □ Gio □ Ven

Scegliere eventuali giorni di chiusura dello sportello

Giorni di chiusura definiti

Data di chiusura	Giorno	Rimuovi
15/02/2011	martedi	🗑

Il servizio **Residenza cittadini comunitari** per il **Settore Anagrafe**

Nessuna richiesta definita

(a)

Stai modificando il passo "Orari di erogazione".

Nella scheda centrale devi definire:
- la durata di ogni appuntamento
- le fasce orarie di erogazione del servizio nei giorni di apertura dello sportello

Premere OK per modificare il passo.

Definisci la pagina degli orari di erogazione della prestazione presso lo sportello

Definire la durata di ogni appuntamento

Durata*: 10

Definire le fasce orarie per ogni giorno della settimana

Lunedi | Martedi | Mercoledi | **Giovedi** | Venerdi

☑ Sportello aperto

Definire la fascia oraria

Nome*: Pomeriggio

Orario di inizio*: 14:00 Ora di fine*: 16:30

Aggiungi

Fasce orarie definite per il Giovedi

Nome fascia	Ora inizio	Ora di fine	Rimuovi
Mattino	8:00	12:00	🗑
Pomeriggio	14:00	16:30	🗑

Il servizio **Residenza cittadini comunitari** per il **Settore Anagrafe**

Giorni di erogazione	1
Orari di erogazione	2
Dati personali	3

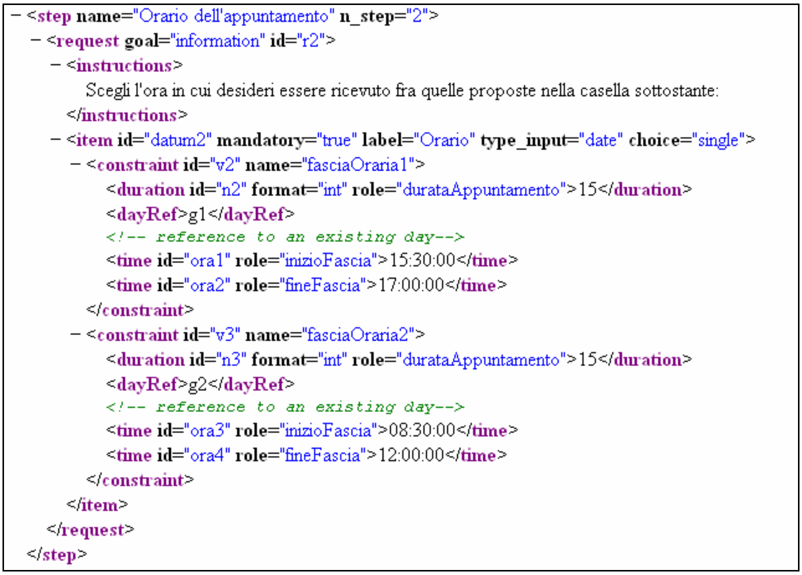
(b)

Fig. 4. The EUD environment for creating e-government front-office reservation services: (a) creation of the date selection step; (b) modification of the step for time selection.

When the end-user developer has created all the steps of the service, s/he selects the button "Completa servizio" (complete the service). The system then generates an XML file, validated against the XML schema discussed in the previous section. This XML file – which is actually the service model – is properly interpreted and used by the EUD environment, along with a fixed style sheet used by the municipality for e-government services, to generate the service itself.

Figure 5 reports a portion of the XML file related to the step for time selection. It actually describes the constraints for the generation of time choices, according to the established appointment duration and time bands, and referred to the week days in which the office is open. Therefore, the element `step` is used (see the XML schema in Figure 3), including an element `request` with the attribute `goal=“information”`. The element `request` includes, in turn, the element `instructions`, whose content (in Italian) asks to choose the hour when the citizen would like to be received, and an element `item` including two elements `constraint`. Each `constraint` element states the duration of appointments (element `duration`) and the time band (elements `time`) related to an open day chosen in the date selection step and referred by means of the element `dayRef`.

The portion of the file shown in Figure 5, when interpreted, generates the web page for time selection like the one illustrated in Figure 1.



```
- <step name="Orario dell'appuntamento" n_step="2">
  - <request goal="information" id="r2">
    - <instructions>
        Scegli l'ora in cui desideri essere ricevuto fra quelle proposte nella casella sottostante:
      </instructions>
    - <item id="datum2" mandatory="true" label="Orario" type_input="date" choice="single">
      - <constraint id="v2" name="fasciaOraria1">
          <duration id="n2" format="int" role="durataAppuntamento">15</duration>
          <dayRef>g1</dayRef>
          <!-- reference to an existing day-->
          <time id="ora1" role="inizioFascia">15:30:00</time>
          <time id="ora2" role="fineFascia">17:00:00</time>
        </constraint>
      - <constraint id="v3" name="fasciaOraria2">
          <duration id="n3" format="int" role="durataAppuntamento">15</duration>
          <dayRef>g2</dayRef>
          <!-- reference to an existing day-->
          <time id="ora3" role="inizioFascia">08:30:00</time>
          <time id="ora4" role="fineFascia">12:00:00</time>
        </constraint>
      </item>
    </request>
  </step>
```

Fig. 5. A portion of the XML file specifying an e-government service

6 Related Work

The Network of Excellence on End-User Development funded by the European Commission during 2002-2003 has defined EUD as “the set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create or modify a software artifact” [18]. This definition implies that the level of complexity of EUD activities may vary in a wide range, from just choosing among alternative behaviors already available in the application (called *parameterization* or *customization* in [17]), to actual programming (called *program creation* or *modification* in [17]) carried out through various techniques, such as programming by example, incremental programming, model-based development.

The latter is the most interesting class of EUD activities, since it should foster end users to perform sophisticated tasks without requiring them to really learn ‘how to do programming'. Therefore, much research work has concentrated on defining appropriate EUD techniques for program creation or modification, also taking into account the application domain. Component-based approaches are proposed for example in the field of computer-supported cooperative work [22][34]. Model-based approaches are adopted in [2][19] to develop nomadic and multimodal applications, and generally support EUD through abstractions meaningful to end users, such as task models [24]. Annotation mechanisms and visual programming are the main EUD techniques implemented in software shaping workshops [5], which have been experimented in the medical and mechanical engineering domain. Visual programming is also proposed in [13], to allow museum curators to create guides for mobile and large screen stationary devices, and in [26], where the approach allows business users to create enterprise widgets. Still within the business domain, Anslow and Rielhe [1] propose to adopt wiki technology as a platform to support end users in the development of business queries. Business process composition is allowed in the Collaborative Task Manager prototype, by means of a programming-by-example technique [27]. In this line, also mashup makers, usually adopting component-based approaches, are regarded as tools for EUD [15], namely for creating systems that support sharing, integrating and searching for information [35]. With respect to these works, the EUD approach proposed in this paper adopts visual programming and model-based techniques in a different way, by customizing the EUD interaction experience to the work practice of the potential end-user developers.

Designing appropriate EUD techniques for a given application domain represents a fundamental challenge that should be addressed, according to Fischer and Giaccardi [8], by adopting a *meta-design* approach. Indeed, meta-design is an emerging "conceptual framework defining and creating social and technical infrastructures in which new forms of collaborative design can take place" [8, p. 428]. By commenting on the concept of meta-design, Sutcliffe and Mehandjiev said: "EUD then becomes a two-phase process: designing the design world, followed by designing the applications using the design world" [28, p. 32]. This two-phase process is evident in our work: actually, the definition of a meta-model for services and the design of an EUD environment for instantiating the meta-model can be regarded as a meta-design activity.

The idea of adopting a meta-modeling approach to end-user development of web-based applications has been proposed also in [7][14]. In these works, the meta-model is based on different abstraction levels, namely a shell level, an application level and a function level. However, all these levels are described according to a software engineering perspective, whereas our meta-model is an abstraction obtained from four different perspectives, according to a multi-faceted approach [11]. As such, it better synthesizes the concept of e-government service to drive the design of an EUD environment for service creation. Moreover, end-user developers involved in the project described in [7][14] interact with the EUD system using SBOML (Smart Business Object Modeling Language), as discussed in [16]. The syntax of this language, even though defined by authors as 'near-English', seems too much computer-oriented, and thus difficult to be learned by domain experts who do not possess any knowledge of programming languages. By contrast, we have designed an EUD environment adopting a metaphor and an interaction style that appear more suitable to the profile of our end-user developers. Indeed, the interaction with this environment has been designed on the basis of the same interaction patterns that characterize existing e-government services and that are usually adopted in this domain (see for example [25]).

The works adopting meta-modeling mentioned above are based on the WebML conceptual modeling language. WebML [4] and its recent extensions (e.g. [12]) is actually very effective for designing and developing web applications from a software engineering and data management point of view. The additional perspectives considered in this paper (HCI and domain experts' perspectives) could probably be integrated in WebML to cope with end-user development problems.

7 Conclusion

In this paper, we have described the ongoing work with the Brescia Municipality to empower administrative personnel to create web-based G2C services.

Although we have not carried out an in-depth experimentation yet, we have collected initial feedbacks from a few employees by presenting demos of the system and asking them to create some new services for booking appointments. Employees have appreciated the system usefulness and positively commented on the easy and engaging interaction experience it offers. Particularly, they have recognized in this application the opportunity of improving the internal procedure currently adopted for service creation, which heavily depends on a complete and unambiguous communication with software developers. Employees also confirmed that a correct mapping is provided between their mental model of how to create services and the conceptual model offered by the application. We have then explicitly asked to these potential end-user developers whether they are willing to include the activity of service creation in their daily work: all of them appeared as high motivated in doing this job, since they judged it very easy and quick; however, to be totally satisfied, they ask for the integration of a preview functionality that shows the service under creation.

As future work we are planning to carry out an extended experimentation with administrative employees to analyze and solve the weaknesses of the developed EUD environment. Accessibility aspects of services will be also considered in the future. Indeed, the EUD environment should support employees in creating web applications compliant with guidelines for accessibility [32][33], so that all citizens can use them, regardless of their cognitive or physical disabilities and hardware/software limitations. This is a very important goal, necessary to accomplish the so-called 'inclusive e-Government' [20]. To this purpose, we will study an extension of our previous work [9][10] to the case of e-government service creation also taking into consideration the design guidelines proposed in [21].

The application will also be extended to support employees to create other classes of e-government services. To this end, we are studying a richer meta-model including specific aspects related to level 4 (Transaction) and level 5 (Personalization) services. A further evolution of the application concerns the integration of functionalities for managing services' input data sent by citizens, thus providing employees with a unique environment for designing online services and dealing with information coming from their actual use.

Acknowledgments. The authors wish to thank Sergio Colosio from Brescia Municipality (Comune di Brescia, Italy), and Alessio Giovanni Cecchi and Federica Santicoli for the development of the EUD environment.

References

1. Anslow, C., Rielhe, D.: Towards End-User Programming with Wikis. In: Proc. WEUSE 2008, Leipzig, Germany (2008)
2. Berti, S., Paternò, F., Santoro, C.: Natural Development of Nomadic Interfaces Based on Conceptual Descriptions. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End-User Development, pp. 143–159. Kluwer Academic Publishers, Dordrecht (2006)
3. Capgemini, The User Challenge: Benchmarking The Supply of Online Public Services, 7th Measurement, Delivered for the European Commission (2007), `http://ec.europa.eu/information_society/eeurope/i2010/docs/benchmarking/egov_benchmark_2007.pdf`
4. Ceri, S., Fraternali, P., Brambilla, M., Bangio, A., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kaufmann, San Francisco (2002)
5. Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A.: Visual Interactive Systems for End-User Development: a Model-based Design Methodology. IEEE Transactions on Systems Man and Cybernetics, part A - Systems and Humans 37(6), 1029–1046 (2007)
6. Costabile, M.F., Mussio, P., Parasiliti Provenza, L., Piccinno, A.: End Users as Unwitting Software Developers. In: Proc. WEUSE IV 2008, Leipzig, Germany, pp. 6–10 (2008)
7. De Silva, B., Ginige, A.: Meta-Model to support End-user Development of Web based Business Information Systems. In: Baresi, L., Fraternali, P., Houben, G.-J. (eds.) ICWE 2007. LNCS, vol. 4607, pp. 248–253. Springer, Heidelberg (2007)
8. Fischer, G., Giaccardi, E.: Meta-Design: A Framework for the Future of End User Development. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End-User Development, pp. 427–457. Kluwer Academic Publishers, Dordrecht (2006)
9. Fogli, D.: End-User Development for E-Government Website Content Creation. In: Pipek, V., Rosson, M.B., de Ruyter, B., Wulf, V. (eds.) IS-EUD 2009. LNCS, vol. 5435, pp. 126–146. Springer, Heidelberg (2009)
10. Fogli, D., Colosio, S., Sacco, M.: Managing Accessibility in Local E-government Websites through End-User Development: A Case Study. Int. J. UAIS 9(1), 35–50 (2010)
11. Fogli, D., Marcante, A., Mussio, P., Parasiliti Provenza, L., Piccinno, A.: Multi-facet Design of Interactive Systems through Visual Languages. In: Ferri (ed.) Visual Languages for Interactive Computing: Definitions and Formalizations, pp. 174–204. IGI Global (2007)
12. Fraternali, P., Comai, S., Bozzon, A., Toffetti Carughi, G.: Engineering rich internet applications with a model-driven approach. ACM Trans. on the Web 4(2) (2010)
13. Ghiani, G., Paternò, F., Spano, L.D.: Cicero Designer: An Environment for End-User Development of Multi-Device Museum Guides. In: Pipek, V., Rosson, M.B., de Ruyter, B., Wulf, V. (eds.) IS-EUD 2009. LNCS, vol. 5435, pp. 265–274. Springer, Heidelberg (2009)
14. Ginige, A., De Silva, B.: CBEADS©: A Framework to Support Meta-design Paradigm. In: Stephanidis, C. (ed.) HCI 2007. LNCS, vol. 4554, pp. 107–116. Springer, Heidelberg (2007)
15. Grammel, L., Storey, M.: An End User Perspective on Mashup Makers. Technical Report DCS-324-IR, Department of Computer Science, University of Victoria (September 2008)
16. Liang, X., Ginige, A.: Enabling an End-User Driven Approach for Managing Evolving User Interfaces in Business Web Applications - A Web Application Architecture Using Smart Business Object. In: Liang, X., Ginige, A. (eds.) Proc. Int. Conference on Software and Data Technologies (ICSOFT 2007), Barcelona, Spain, pp. 70–78 (2007)
17. Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-User Development: An Emerging Paradigm. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End-User Development, pp. 1–8. Kluwer Academic Publishers, Dordrecht (2006)

18. Lieberman, H., Paternò, F., Wulf, V. (eds.): End-User Development. Kluwer Academic Publishers, Dordrecht (2006)
19. Manca, M., Paternò, F.: Supporting Multimodality in Service-Oriented Model-Based Development Environments. In: Forbrig, P. (ed.) HCSE 2010. LNCS, vol. 6409, pp. 135–148. Springer, Heidelberg (2010)
20. Millard, J.: Inclusive eGovernment: survey of status and baseline activities. Prepared for the Inclusive eGovernment Expert Group of the European Commission, pp. 1–54, http://ec.europa.eu/information_society/activities/einclusion/library/studies/docs/inclusive_egovernment_survey_12_07.pdf
21. Money, A.G., Lines, L., Fernando, S., Elliman, A.D.: e-Government online forms: design guidelines for older adults in Europe. Int. J. UAIS 10, 1–16 (2011)
22. Mørch, A., Stevens, G., Won, M., Klann, M., Dittrich, Y., Wulf, V.: Component-Based Technologies for End-User Development. Communications of the ACM 47(9), 59–62 (2004)
23. Palvia, S.C.J., Sharma, S.S.: E-Government and E-Governance: Definitions/Domain Framework and Status around the World, Foundation of e-government. In: ICEG, pp. 1–12 (2007)
24. Paternò, F.: Model-based design of interactive applications. Intelligence 11(4), 26–38 (2000)
25. Pontico, F., Winckler, M., Limbourg, Q.: Organizing User Interface Patterns for e-Government Applications. In: Gulliksen, J., Harning, M.B., van der Veer, G.C., Wesson, J. (eds.) EIS 2007. LNCS, vol. 4940, pp. 601–619. Springer, Heidelberg (2008)
26. Spahn, M., Wulf, V.: End-User Development of Enterprise Widgets. In: Pipek, V., Rosson, M.B., de Ruyter, B., Wulf, V. (eds.) IS-EUD 2009. LNCS, vol. 5435, pp. 106–125. Springer, Heidelberg (2009)
27. Stoitsev, T., Scheidl, S., Flentge, F., Mühlhäuser, M.: Enabling End-User Driven Business Process Composition through Programming by Example in a Collaborative Task Management System. In: Proc. Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2008), Herrsching am Ammersee, Germany, pp. 157–165 (2008)
28. Sutcliffe, A., Mehandjiev, N. (Guest Editors): End-User Development. Communications of the ACM 47(9), 31–32 (2004)
29. Tidwell, J.: Designing Interfaces – Patterns for Effective Interaction Design. O'Reilly, Sebastopol (2005)
30. Tidwell, J.: Common Ground: A Pattern Language for Human-Computer Interface Design, http://www.mit.edu/~jtidwell/common_ground.html
31. Xu, Y.: Electronic Service Delivery: Endeavor to improve the Government Innovation. In: Proc. ICAMS 2010, Chengdu, China, pp. 105–109 (2010)
32. WAI-ARIA, Accessible Rich Internet Applications (WAI-ARIA) 1.0, W3C Working Draft, September 16 (2010), http://www.w3.org/TR/wai-aria/
33. WCAG 2.0, W3C Recommendation, December 11 (2008), http://www.w3.org/TR/WCAG/
34. Wulf, V., Pipek, V., Won, M.: Component-based tailorability: Enabling highly flexible software applications. Int. J. Human-Computer Studies 66, 1–22 (2008)
35. Zang, N., Rosson, M.B.: What's in a mashup? And why? Studying the perceptions of web-active end users. In: Proc. Int. Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2008), Herrsching am Ammersee, Germany, pp. 31–38 (2008)

From System Development toward Work Improvement: Developmental Work Research as a Potential Partner Method for EUD

Anna-Liisa Syrjänen and Kari Kuutti

Department of Information Processing Science
University of Oulu, Finland
P.O. Box 3000, FI-90014 University of Oulu
{anna-liisa.syrjanen,kari.kuutti}@oulu.fi

Abstract. The dominant strategy within the field of EUD has been to improve end-user activities within a single software system. This approach has some limitations. First, the work environment often consists of a number of different systems and tools that form an information ecology with which users must cope. Second, the use of computers is embedded in organizational practices that may also need to be changed. Thus, there is a need to combine EUD with a parallel development of work practices. However, common work-development approaches, for example, process improvement, usually adopt a top-down managerial point of view that relies on expert modeling, and are therefore incompatible with EUD ideas. This paper suggests that a work improvement method developed in Finland since the 1980s, developmental work research, is a good candidate for partnering with EUD, because it takes the potential of local grassroots innovation and the development of work practices seriously.

Keywords: Systems Design, End-User Development, Developmental Work Research, Information Ecology, Workplace Technology.

1 Introduction

End-user development (EUD) has a long history reaching back to the 1970s, with roots both in practice among the multiuser mainframe communities and in research on how to use emerging information technology (IT) or interactive workstations in the Xerox PARC. Part of the seminal work in the field was done, for example, by Bruce I. Blum [1], James Martin [2], and Allen Cypher et al. [3]. During the 1980s, research and development were active, involving scientists that could be called end-user developers, too; but the field was hit heavily by the rapid emergence and acceptance of personal computers (PCs) and the "productivity tools." These partially addressed the same problems that EUD did (although in a prepackaged form) and were capable of delivering "computing power to the masses"—very successfully, in fact, in the case of spreadsheets. Thus, during the 1990s, the EUD field itself was rather dormant.

However, during the same time, ethnographically oriented workplace studies (e.g., [4], [5], [6], [7: 18]), initiated by the rise of computer supported cooperative work

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 123–138, 2011.

(CSCW) research and related to the use of IT in work practices, started to become more common. These studies have helped to recognize the problems in IT use that neither system design nor productivity tools alone seem able to cure. Here, EUD may still have a place and a role to fill. This, and the improvement of technological possibilities ensuing from the rapid development of IT, with respect to processing power and communication capability, led to a renewal of the EUD research field during the first decade of the 2000s.

A common approach in current EUD has been to provide a "front end" to an existing system, one that is easy enough to use and powerful enough that, with it, end users can tailor and to some extent reprogram the outputs of the system using the tools provided and the data available in the system. For instance, in [8] the visual generalization idea, what the user can see, epitomizes what should be programmable (p. 372), and it is seen as the central problem of example-driven programming (p. 2, 6). Were it to be solved, programmable systems for ordinary end users could also be produced. The most challenging future areas of EUD are thus the novel interface technologies such as the integration of context awareness and adaptability, the sharing of EUD artifacts and repositories with recommendation support, and guidelines for flexible component-based systems [9] (p. xii). Lieberman et al. [9] see the ultimate goal of EUD as being to invisibly support end users' software development so that they can use domain-specific "know-how" to shape their new IT tools (p. 485).

While this approach can address a number of issues, it still has a couple of major limitations with respect to the common needs of the situations in a workplace. One is the increasing recognition that the thinking workplace is technologically homogenous, which is a rather strong simplification. Actually, workplace technology is a collection of various systems, services, and tools. These have perhaps never been designed to form a coherent whole; but they have emerged in the course of work as partial solutions to acute problems, and they now form a sort of information ecology with which users cannot but try to cope as best they can. One example is the current way of introducing social media, per se, as tools for organizational discussion. Quite soon, the user communities will be headed for social computing, which entails a grasp of the old discourse, strategic organizational development, classical work practice, and systems development with webs of technologies that may need to change.

Related to end user-driven development efforts, the concept of information ecology has received some attention [10], [11], [12]. In adopting such an ecological IT view, prior to employing the "know-how" aspects, one should also be capable of analyzing the "know-why" issues, which help to shape ideas about how development could be supported technologically. Unfortunately, practical design cases where the idea could have been addressed are scarce; only the particular end users can see what makes the system, product, or service useful [10] (p. 65) as part of their workplace technology.

Another and maybe even more fundamental problem of looking at EUD from the point of view of only one IT system is the fact that system use does not normally constitute the entirety of work, often not even a major portion of it. Rather, it is just an element of a wider network of individual work practices and social and organizational procedures, rules, and conventions. Often a change in an IT system will provide only a marginal benefit if the related work procedures and practices remain the same. It is clear that some development of work takes place with every application of EUD, but there is the danger of a certain technocentrism, if the situation is seen only through EUD “eyeglasses.” To be balanced and efficient, the scope of such efforts should be broadened and end-user system development should be complemented by end-user driven development of work. For analyzing work, however, conceptual tools are needed that are different from those for system EUD, where the focus naturally is on data and data processing. Even in those system design methods where the context of the system’s use is taken seriously (e.g., contextual design [13]), there is no guarantee that the “rest” of the work will be treated comprehensively. Therefore, it is necessary to complement system design methods with work design methods. But there is a catch: the commonly used work design methods adopt a managerial, organizational-effectiveness perspective, and they rely heavily on expert modeling of the work and the organization [13], [14], [15], [16], [7](p. 88).This would be in direct contradiction to EUD’s grassroots ideals, and it leads to a tension that cannot be easily reconciled.

There are, however, certain work development methods that take the potential of local grassroots innovation and the development of work practices seriously. The purpose of this paper is to show one work improvement method developed in Finland since the 1980s, developmental work research (DWR), which seems a good candidate to be used with EUD. DWR has been widely used in a variety of organizational settings, both in the public sector and in industry. It has rarely been combined with IT development, however, and probably just for the reason that it does not fit well with the normal top-down expert modeling approach used in system design.

We start our review with historical end user-driven systems development and in situ field studies on programming and work (chap. 2), and continue by approaching aspects of ecological technology (chap. 3). We then introduce DWR (chap. 4) and conclude (chap. 5) by suggesting that end-user system development be complemented with end-user driven development of work.

2 From an EUP System to in Situ Systems Design

The leading idea within the field of EUD has been to improve end-user activities within a single software system by providing tools for end users’ programming that automate some of the tasks. This kind of development has mostly focused on programming technology, and the corresponding use of EUD tools has mostly been close to actual end-user programming (EUP). This feature is still present in a number of current EUD approaches, and it is naturally important when the automation of tasks is involved. As shown by ethnographic field studies on work practices and studies on science and technology [4], [5], [6], [7], [17] and implementation [1], the work practices of end users are considerably broader than what can be taken into account by the current EUD perspective. The reciprocal relationships between use, other work practices, and design changes stimulate both domain knowledge and the development of the work community. This perspective should thus not be reduced to operational efficiency, as still is often done in information technology marketing. It would be beneficial to adopt a broader perspective, and EUD should be embedded in the development of all work practices and organizational procedures.

2.1 Approaching End-User Programming

Initial steps toward EUP had already been taken during the mid-1970s, and two different sources for them can be identified. One was directly oriented toward practices in organizations, which emerged within interactive, multiuser, mainframe communities as a response to the programming bottlenecks and slow system-development cycles of that time. The hope was to alleviate these problems by changing the role of end users to that of autonomous programmer [2]. Another, more ambitious and future-oriented step took place during the development of the first personal workstations at Xerox PARC [18].

The first EUP step related to the use of existing, widely available technology, and as James Martin's book testifies, it had gained a certain practical foothold by the early 1980s. It could have developed into a significant movement, because it was backed by major vendors such as IBM. However, its advance was heavily disturbed by the emergence and rapid acceptance of the PC and its productivity programs that at least partially fulfilled the same role. Toward the end of the 1980s, PCs had won the day, and EUP in the mainframe-computing environment was sinking into oblivion.

The other, more research-oriented direction in which interfaces had a significant role did not suffer so heavily from the impact of PCs. User-friendly design solutions, such as metaphors and analogies, were adopted for the design of end user-driven programming systems. Personal workstations (and PCs later on) emphasized the purpose of work and the end users' interest in using computers by programming them to do useful tasks, such as accounting and macro-programming with spreadsheet applications. End-user programming and software tools designed for programming were promoted in the spirit that "programming could be good for people" [18] (p. xii). This strategy was criticized, but regarding EUP it was also significant, as the potential of programmable IT was designed for actual end users, and demanded the ability to apply different approaches to each use, design, and many other activities, in relation to the everyday, changing application domains.

One of the major approaches has been programming by demonstration (PBD) [3], which later became known as programming by example (PBE) [8]. The ideas of a software agent recording a user's acts and making a new program based on the user's demonstrated acts were proven work [8] (p.2). A number of EUP systems were developed based on the ideas and a number of experimental projects that were started (e.g., research on iconic programming, visual programming, direct manipulation interfaces, construction kits, and script languages). However the non-IT-educated and end users not having high level domain skills could not apply the systems and learn the programming as expected [8](p. 5, 132, 160). This might have resulted from inappropriate or lacking user studies [18]. According to Martin [2] a number of end user-driven systems were created without actual systems analysts (p. 161).

Martin [2] has also pointed out that analysis has always played a highly valuable role, even when end users are creating their own applications; in most software corporations, this demanded a total change, not only in systems analysts' jobs, but also in the ways of cooperating with end users (p. 322). The PBD and PBE ideas have thus been very important in promoting programming as a useful activity for end users, and because of the conceptual development within the relationship between use and programming started. For early developers of the EUP systems, a difficult task was and still seems to be the “escape from earlier disciplines” [2] (p. 333). Defining the end user’s programming or design [18] without using technical terminology or with terms for the user’s tasks [8](p. 132) is challenging, especially within the actual end users’ context of work. Research projects have attempted to alleviate the problems, but the problem related to understanding the actual context of end users’ work is still open. In addition, the interest in developing and producing end user-programmable systems has been languishing since the 1990s, from one thing or another.

The specialization of professions, politics, and business, and other societal changes, influenced the development of EUP systems [17], [18]. Alan Kay also listed several factors related to end users’ context-of-use, such as end users’ goals, varied intentions, concepts, and subject matter, which had been seen as vague or alien, and thus less informative for the early developers. Even in cases where it was realized that EUP might support the dynamics of end users’ activities [18], solutions did not deliver the level of service required. This stimulated changes in research methods, so that in situ field studies on EUP started to emerge (e.g., [4], [6], [7] p. 18).

2.2 Approaching the Work Practice

Since the 1980s, the increasing influence of anthropology and microsociology has brought field studies of work practices into IT research. In one ethnographic research study on end-user computing [6], Nardi discusses a new vision of end users’ work, in which personal computers are not only used for certain purposes, but are also programmable with the end-user programming facilities present in the systems. Exploring task-specific programming and application-making, she criticizes the narrowness and one-sidedness of the job descriptions that were used as the basis of many early EUP systems (p.10). More often than not, users’ “work” has been defined in extremely narrow terms, such as single operations, tasks, actions, or interactions with a particular application, computing, or programming tool.

According to Nardi [6], systems developers lacked an understanding of the social and dynamic aspects of end-user programmers’ work practices. The developers could not see the contexts (e.g., cooperation among users; dynamic shifts between use, design, and other activities; the interrelated sets of use and design tools; the designed, ad hoc, and self-made solutions; the prototypes, examples, customizations; or the tailoring of solutions) that were relevant to making a new application. Instead of looking at only individual tasks, it is necessary to see that the context of work is an intertwining of design, use, and other related activities. To preserve the design idea [19], short-term factors should be seen as part of the wider, historical work activity. Otherwise, the kinds of dynamic shifts in the end users’ activity become interpreted ahistorically (e.g., [20]).

Although ethnographic field studies did not directly result in actual new EUP systems, important issues for EUD did emerge. While, on one hand, the knowledge and skills needed in design have been proven to be self-learnable, and end-user designers have been shown to be capable of training themselves in the context of use [1], [20], on the other hand, highly specialized technological or IT domain knowledge has generally been seen as an obstacle to actual user involvement in design [14], [21], [22], [8] (p. 132). This has led to mediated use or work practices [23], surrogate user profiling [16], and diversification of the professional programming tools and EUP systems.

Working with users demands special skills [24], such as an ability to grasp the "invisible" [25]—the domain that is open and shaped for each implementation [1]—and the organizational procedures and work practices that need to change [19]. The in situ field studies have helped us to see the problems in IT use that neither systems design, productivity tools, nor improved systems analysis [2] (p.339) seem able to cure. When EUD is extended beyond the common user-friendly, single-solution EUP, the gap between use and work practice will be more visible, and we shall see how the end users' situation and programming differ from those of professionals.

Authors have tried to stress the practice known as computing [6], design in use [4], or co-design [20], [26], just to mention a few of the many varieties. This variety means that we have to be careful in approaching the use, design, and entire scope of practical activity that shape end users' everyday domain. Work practice is not only a matter of defining the context of use or the application domain. Above all, it is a question of the diverse needs of people, within a joint or collective activity [27]. Thus, a change in an IT system, either by professional or end-user programming, will give only a marginal benefit if the related work procedures in practice remain the same. Correspondingly, if work procedures and practices are changed according to a top-down approach, local development possibilities cannot be taken into account.

As a result, we have to find a way to broaden EUD for work, while maintaining its grassroots ideals, so that people can build their own IT effectiveness locally when designing systems in situ and improving their work as needed.

2.3 Approaching the Ecological Systems Design In Situ

Based on their in situ systems research studies, Nardi and O'Day [10] are convinced that technology can be put into practice according to local practices and values, but that development of practices to fit a given technology is not the only solution (p. 211). They believe that it is important to maintain possibilities for end users to adapt their IT according to the dynamics of the work and the desire to reflect human values. Their information ecologies approach describes EUP as a long-term collaborative activity to develop an information system (IS) to meet the needs of work. Programming systems in situ and other users' longer-term participation in design are consequently fundamental.

The important point here is that there are not only "power users"—very experienced users who are technologically oriented or domain expert users [9](p. 25, 116, 357) and who use technology effectively and responsibly [10](p. 211). There is also a full range of other kinds of users with varying skills and motivations, who in practice possibly have to be taken into account somewhere along the line, in order to achieve the changes needed. Hence, the important question is whether we see only the IT tool or system at hand and ignore related social, knowledge, or other systems used within the community of users. If we ignore these invisible yet meaningful work systems, end user-driven programming departs from its grassroots ideals. If we take these systems into account and adopt an ecological, long-term perspective that enables us to see the development ideas of local people, end-user system development can be complemented by development of work [27], [28], [29],[30].

The most instructive case of EUP within work is the classic study by Blum [1]. He carried out a long-term, in situ, systems-design effort by which end users having no computer background acted as programmers and knowledge developers. Blum's grassroots ideals for end user-driven systems development in situ [1] were based on the openness of the software product, the technological and domain (i.e., healthcare) knowledge (p. 257), and a 14-year analysis of the evolving, in situ IS designed as a work system. Rather than assuming that end users intimately know their domain and lack the skills to adopt technological or systems-development knowledge, Blum analyzed their work in situ, and the conceptual model and the system were developed for this particular activity. They represented the evolving knowledge of interest in conceptual structures as being as complex as the evolving understanding about that domain (p. 306), in the context of clinical cancer research.

Blum's seminal work on programmable information systems has, unfortunately, not been widely recognized. He was able to raise a number of issues about end users' activities related to EUP and EUD that were later rediscovered by field studies or were discussed again in later research (cf. [6], [8], [9], [20],[31], [32]). Unfortunately, quite often discussions have been limited to data processing or IS development, and only rarely have they touched on the knowledge or developmental work of the local end users, as in Blum's original work. Quite often, the grassroots perspective on development has been entirely missing, or it has been based on short-term user studies leading to oversimplified end users' views regarding their domain knowledge. However, for EUD and in situ systems design, we have to emphasize the importance of knowledge to an end user's actual IT strategy [33], [34].

Systems design in situ is already recognized as a design and research topic [4], [9], [17], [35], but its progress has been compromised by the lack of end user-driven programming tools and programmable systems. The inability to develop such tools may be due to a lack of understanding of the nature of programmability in the end users' everyday work and domain and of what could be seen as development of work [36](p. 164). Because of the dynamic features recognized in field studies of work, it is necessary to take time into account as an important dimension of EUD. This requires programmability, which correspondingly should be seen as domain or work oriented, rather than only as a technology-oriented or design-oriented issue. For the in situ systems-design approach to EUD, it is necessary to strive toward making the programmability of such systems applicable and maintainable within the end users' everyday work and domain. A better understanding of programming and use practices can finally lead to the emergence of work-driven systems' programmability, through programmable IS development, which merges with the work activity of end users and provides ideas for new practices and IT support when needed.

3 Ecological Views on Technology

Application software usually prescribes the procedures for using it and greatly restricts the possibilities for end users to make changes. When we use these systems to solve our real-world problems, whatever we do must follow the procedures and pathways programmed into the tools. In order to introduce new procedures and make changes in the tools used by user organizations, end users currently have two main options. First, end users can be consumer-users [35] (p. 432) and wait for the next version of the system, in the hope that it will respond to their needs. Second, end users can be involved in a professional system-development set-up [13], [16], [24], by which a new system may become available over time, possibly even years later. Either of these options cannot support the end users' diverse needs to develop work in situ [36], [34], [37]. In addition, the productivity of IT-supported work decreases as users have to invent how to circumvent the obstacles raised by aging technology.

This situation and a recognition of the many technological possibilities of programmable IT [1], [2], [3], have revitalized the EUD field [8], [9], [10]. We should, however, be aware of the problems encountered historically, so that new ideas do not repeat the same kinds of problems time after time: providing ideas only for narrow front-end EUD or systems; applying overly naïve user friendliness or simplifications by UCD or PD, which just alienate end users from the technology; and thinking of workplace technology as homogenous and of EUD in terms of only one IT system being in use. We should thus avoid the inherent technocentrism in approaching the situation of end users when looking only through EUD "eyeglasses", and be able to balance the need for new IT and the development of work, so that end-user system development can be complemented by development of work.

The above goals fit work-oriented EUD, and in practice, they point out that instead of focusing on a systems use, IT systems, and tasks per se, we need to study the work practices, social and organizational procedures, and rules and conventions that have shaped the existing workplace technology. This technology, consisting typically of various systems and tools, has rarely been designed to form a coherent whole, but has emerged in the course of history as partial solutions to acute problems solved locally, so that users could do their daily work. It can be seen as a sort of information-based and work practice-based ecology, where technology and practices are adapted to fit well enough with the local situation [10] (p. 211).

We shall next introduce some ecological, or more-heterogeneous, approaches related to the workplace technology and grassroots activity, which aim toward an appropriate, evolving coherence to IT in everyday life.

3.1 Information Ecology

Ethnographic workplace and technology studies [4], [5], [6], [7] have revealed the complexity related to IT use in the workplace. Nardi and O'Day [10] introduce the concept of "information ecologies" to stress the dynamics of how end users make IT manageable as part of their work (p. 65). Thus, by adopting an ecological point of view, one should also be capable of analyzing the "know-why" issues and not only the "know-how" issues that have been the interest of many EUD developers.

The authors [10] differentiate between seeing the technology as a tool, text, system, or ecology (pp. 25–30), terms that all reveal certain facets of understanding IT in use and the assumptions taken for granted when talking and thinking about IT in everyday life. When we use a tool metaphor, we see IT as useful for the task at hand and for the user controlling it. When a tool is integrated with the context of use, it is important to choose the right tool, learn to use it, achieve skill at its effective use at work, and finally discover its limitations. Designing a good tool is challenging, because the utility, usability, skill, and learning demands require designers to anticipate the users' needs. The authors underline that tools should also allow a certain amount of freedom to modify the IT when needed. [10].

Technology seen as text suggests that users should be capable of reading the inscribed meanings of the IT designed by other people who are not users [10]. Many IT problems ensue from the distance between use and design and different cultures, which demands that we also consider multiple ways of doing things, talking, and communicating about the invisible work [25]. This is not a new perspective: Martin [2] already described this as "an impenetrable wall" between the user and the designer who uses traditional methods and is kept away from the end user (p. 77).

Nardi and O'Day [10] view the metaphor of technology as system as the richest and also most troubling and mind-altering perspective (p. 33) as it implies broad and vague issues at different levels of activities. The authors argue that this view negates distinctions among different local settings, and that when systems are observed in detail at the system level, the effect can seem overwhelming (p. 47). Thus, the authors turn to the information ecologies metaphor, which epitomizes local people, practices, values, and technologies maintained by members of the collective.

The point made by information ecology is that instead of worrying about the wide generality of predefined end user-driven systems [3], [8], the systems ought to be open, programmable for the full range of activity of the actual end users, for their needs. These values are the key to developing systems in situ. Such a design perspective is close both to Blum's design ideas and Fischer and Giaccardi's meta-design framework, in which end-user development plays a future needs-oriented role.

3.2 Coping with Complexity

The "ecological" complexity of workplaces has been addressed already by Blum [1] who emphasized the importance of the environment where systems are initiated, used and evaluated. According to him, we should analyze and accept the situatedness of the environment that influences our activity (p. 101). We thus should produce open systems, that is, software systems that can be modified by end users when work changes, and which evolves in terms of collaboration, interaction, and reflective deliberation as a means of improving end users' work resources. This moves our mindset from the "plan-and-build paradigm" to the "build-and-evolve paradigm" (p. 378), as the foundation of programmable EUD systems goes beyond programming.

Thus, according to Blum [1] design implies a looking-forward activity, in which we consciously alter our environment by seeking and working for the identified needs (pp. 102–4). These are not always well defined, may conflict, just as tensions, differences, and competing issues cannot be avoided in human behavior. In fact, they are actually triggers and drivers of development. Avoiding disagreements only results in fruitless searches for unobtainable solutions, and increases non-productivity at work and in designs. Hence, the practice of design means understanding the people who play diverse roles, respond to breakdowns, and solve problems in unique ways.

Accordingly [1] human activity therefore epitomizes the point that expertise is more "arational" than rational, and is based on learned ecological and cultural responses (p. 127). As a consequence, systems must be open, not only for individual action, but also for the work community's activity and dynamic processes. In line with Blum, this equally means that analysis must be completed at the level of the activity, and that flexibility in end users' experiments must be maintained (p. 149).

Recently, Fischer and Giaccardi have expressed similar ideas with their meta-design framework [35]. Like Blum they see the role of designer users as vital. They also argue that people do not play or define use, design, or any other roles as such, but rather dynamically change from one activity to other. Thus, people should be supported in their changing roles, including the full spectrum—from being passive consumers of IT to being domain designers and meta-designers.

Fischer and Giaccardi's domain-oriented design environment and Blum's fragment-based environment implemented through adaptive design are two architectural solutions for open, programmable EUD systems. Both frameworks contain the necessary application domain-driven programmability and support for conceptual design levels, which are familiar to actual end users. They also share the idea of being multilevel in analysis, which in meta-design refers to end-user development for future needs, participatory design for problem framing and solving, and designing "the in-between" for socio-technical integrations. These levels provide the needed openness in the epistemological/computational, social/cognitive, and cognitive/social dimensions that frame the meta-design's design space. [35] (p. 453).

Yet neither of these authors has provided any guidance for the development of end users' actual work practice, beyond the modifications of a system. They do recognize the need to continuously develop work in situ (e.g., to support end users' changing roles) in participative ways, but they do not offer the actual means for doing it.

3.3 Work Development

Selecting a partner methodology to complement EUD is by no means simple. Historically the most common approach to the development of organizations and work processes has been top-down, large-scale rationalization, either as an incremental homogenization by standardized best practices, as in process improvement (PI; e.g., [38]), or a radical reorganization, as in business process re-engineering (BPR; e.g., [39]). Both approaches are based on a managerial perspective and its values, and they apply expert modeling of activities, leaving very little room for local problem solving or innovation. Because of this top-down orientation, it is hard to think that they could be adopted, for example, for social media solutions or combined with EUD, which epitomize opposing values and approaches.

In Europe, there is a long tradition of local grassroots-level workplace-development projects and a number of variations on action research, by which an outside actor acts as an involved mediator/facilitator. In Scandinavia, such an approach is often called "democratic dialogue" [40], and it comes much closer to the EUD ideology of local development than do PI or BPR. Still, it is recognized that the action research process has difficulty finding integrative development directions and implementing larger changes, because it is informal and opportunistic [41], is often based only on everyday experience, and is lacking in analytical tools.

To be able to direct EUD efforts efficiently enough, the work development effort needs a larger scope and possibilities to reflect work collectively. However, attempts at conceptualizing this domain for EUD thus far have been limited to a single design perspective, be it that of the actor, tool, task, place, or abstraction level, and this is not enough. What is still needed is an integrative work and development perspective capable of guiding technology development, as well. In the next section, one such approach, called developmental work research, will be introduced.

4 Developmental Work Research

DWR is a set of methods developed by a research group led by Yrjö Engeström at the University of Helsinki since the 1980s [36]. DWR is an interventionist method for development of work, and it has served both as a concretization and a test bench for Engeström's theoretical ideas on cultural-historical activity theory (CHAT) [27]. According to Engeström, before the introduction of DWR ideas, there had been four main historical phases in general work research [42]. Traditionally, the subject of work research was the rational and effective work done by existing or ready-made systems, with its focus being on how to adapt people to technological and other related work systems. The second, administrative, phase introduced statistical work measurement methods, official proficiency requirements, and the hierarchical positioning of work. The third phase, considered reconstructive work research, was interested in industrial safety. It attempted to ease disproportionate workloads and prevent the dehumanization of work. The fourth phase involved the type of action research that took everyday practices for granted. In such cases, researchers might have revealed defects in work, but without giving actors the means of influencing how work could be improved and resources allocated.

Engeström suggests DWR as the fifth phase, seeking to offer both enduring support and the means of development [36]. The purpose of DWR is the collective transformation and development of work, technology, and organizations (p. 9). It provides the means for goal-oriented, work-oriented, bottom-up development using the ideas from CHAT. This means that work and its development take place within activity systems and certain contextual structures, but are also seen as dynamic and evolving. Because of this work orientation, these ideas have evolved into grassroots practices that improve and maintain the orientation toward future work by promoting social innovation and a learning culture [43].

Based on these grassroots activity ideals, we believe DWR is a potential method for conceptualizing the development of work for EUD. It is a practical methodology, being applied since the 1980s and dozens of concrete cases can illustrate its aspects; for example [36] (pp.199-449], [29], and [43] were PhD projects, but the later use of DWR has expanded beyond the original research projects. It is currently regularly used by various organizations, many of whom work in public sector development.

4.1 Participatory and Interventionist Approaches

DWR [36] is a development strategy using a participatory and interventionist approach (p. 150). The former means that people who are involved in the work analyze and reshape their work themselves. The latter means that ready-made solutions are not brought in from the outside, but tools for analysis, the transformation of work, and the design of systems are done in situ within the work community.

According to Engeström [36], the core concepts of DRW are the historical analysis of work using an activity system as a unit, the object of an activity, the multiple voices of participants of an activity, and the contradictions within the activity situation (pp. 29–44). An activity system is by definition a collective and multi-voiced creation; it includes individuals who are members of their everyday community and the different viewpoints of the various participants, as seen against their personal background. The scrutinized activity [43] is a source of new models, ideas, and decisions for development, which are searched for and negotiated collectively. Thus, results appear in the form of expansive transformation ideas through the collectively identified "zone of proximal development" [36] (p.65), the work community's potential development directions. Both the "vertical" movement across activity levels (individual actions vs. collaboration) and the "horizontal" movement across the boundaries of different work practices are seen as both possible and important (p. 43). In other words, not only do individuals achieve new abilities, but they also learn to change the work with other relevant actors.

DWR helps people learn how to deal with the object of work, which is elusive yet tangible, fragmented yet recollected, including longer-term planning and development, as well. For workers and users, this may be the "rediscovery and expansion of use" value of the object which should stimulate work and maintain productive activity over the entire participating work community. [36] (p. 10).

4.2 Change Laboratory

Today the main method used in DWR [36] is called Change Laboratory® (pp. 291–305), [43], [44]. As the name indicates, it is based on a laboratory, a separate space at the workplace reserved for development activity. The idea is that the laboratory space is separated from the actual work, but is also physically very close to it, to maintain the daily connection. It can immediately be used for reflection, reference, or spontaneous meetings without any further preparations. The space is used by a natural work unit or team, typically helped by a facilitator, at least initially. It contains a set of instruments both to analyze the problems, disturbances, and ruptures of daily work, and to develop models and ideas for new work practices. The approach can be used both for radical changes and for incremental improvements in the work.

The methodology used in Change Laboratory is based on the CHAT-derived notions of remediation and dual stimulation. (Note that [45] contains a good discussion about the theoretical background of the methodology.) According to Engeström et al. [44], it is based on four basic ideas:

- "There is a need for bringing work redesign closer to the daily shop floor practice while still keeping it analytical—a new dialectic of close embeddedness and reflective distance.
- There is a need for bringing together practice-driven redesign of processes and idea-driven construction of visions for the future—a new dialectic of specified improvements and comprehensive visions.
- There is a need for bringing together multiple parallel rhythms of development in work—a new dialectic of long, medium, and short cycles of innovation and change.
- There is need for bringing together the tools of daily work and the tools of analysis and design—a new dialectic of instrumentalities." (p. 2)

A Change Laboratory project typically consists of three phases, which in practice may not be very clearly separated, but proceed like qualitative research in general. Iterations and reorientation may be needed, based on findings as the process goes on. Initially, there is a preparatory phase, wherein representative recordings and samples of current practices are collected by means of ethnographic fieldwork, interviews, customer feedback, statistics, and so forth.

The second phase is analytical, in which a common conception of the current situation and its dynamics are developed by using a two-pronged approach. On one hand, a historical analysis of the development of the particular work practice is developed using activity-theoretical models as tools. The recent history of the practice is studied carefully. The assumptions are that current practices have earlier been fully adequate, and that if there are now problems, they might have been caused by some internal or external changes, the identification of which is very important to understand. On the other hand, current practices, and in particular recorded problematic situations under current practices, are analyzed and played against the historical theoretical models, both to identify the potential causes behind the found problems and to check the realism of the models created. Eventually, a common analytical view of the current practice solidifies, identifying both the sources of current problems and the dynamic forces influencing the practice.

The third phase is for visions and innovations based on the assumptions that solving only current problems is too shortsighted, and that a development process should also try to anticipate what could happen in the near future and then adopt the practice for that contingency. Thus, the development dynamic of the practice is projected toward the future, taking into account what can be assumed about the development of the context and the society in general. This projection is then used as the starting point for various innovations in the work practice, either for solving current problems or anticipating future situations.

In contrast, for example, to project [13], to research-driven PD [10: 44], and to use-design mediation [23], DWR seeks to promote people's own activity, and their self-motivated development in the domain. Through a sustainable participatory strategy—people analyze and develop their work systems in situ—DWR can provide useful conceptual, analytical, and structural tools for work- and domain-driven EUD. Ethnography [5] is typically applied in workplace studies, but it is not the only possible method for generating an intimate understanding of work.

During the research process, participants (including researchers) learn and shift dynamically from one activity to another, being active as interventionists, collectors of data, observers, analyzers, and interpreters of the material. This shapes the development cycle and produces concrete results for transforming the work at hand, while the change is also historically, practically, and collectively grounded. The ongoing process and the reflection cycle evolve together as conceptual tools and sustain the emerging new activity. Ideally, realizing this would require dynamic, modifiable, programmable, and open IT systems—just the kinds systems EUD is striving for.

5 DWR: A Potential Partner Method for EUD

The use of computers is embedded in organizational procedures and work practices, and these may also need to be changed. Thus, there is a need to combine EUD with a parallel development of work practices. However, common work development approaches usually adopt a top-down managerial point of view and rely heavily on expert modeling, and are thus incompatible with EUD's grassroots ideals.

We have argued earlier that EUD would need to be complemented by a development of work perspective, one that would be compatible with EUD ideals and enable the necessary scope and iterations for improvement. DWR seems to be a good partner candidate in this respect—and vice versa: EUD can offer an IT development approach for DWR, which may have been missing, because traditional IT development, with its prefixed requirements, does not easily fit with DWR processes.

When developing EUD together with work aspects, the approach used should contain practical tools and conceptual support for reflection from an individual member, work group, or collective viewpoint. DWR possesses precisely such a toolkit. Therefore, we suggest that EUD should be integrated with a DWR work-development effort and applied to a wider perspective than merely the development of technology. This means refocusing EUD toward information ecologies through development of work. Grounding systems development in situ requires a strong understanding of actual work practices, which calls for field study and intervention-oriented methods. DWR has been developed for work environments consisting of a number of different systems and tools, by which sustainable information ecology can be shaped, and in which end users can cope.

Our conclusion is that the DWR work improvement method is a good candidate method for use with EUD, because it takes the potential of local grassroots innovation and the development of work practices seriously. In the end, the only resource that can improve IT practices involves end users continuously designing systems in situ. We hope we have provided a clear idea of EUD and DWR integration, and we suggest moving it forward throughout the EUD field.

References

1. Blum, B.I.: Beyond Programming. To a New Era of Design. Oxford University Press, New York (1996)
2. Martin, J.: Application Development without Programmers. Prentice-Hall, Englewood Cliffs (1982)
3. Cypher, A., et al. (eds.): Watch What I Do: Programming by Demonstration. MIT Press, Cambridge (1993)
4. Allen, C.: Reciprocal Evolution as a Strategy for Integrating Basic Research, Design, and Studies of Work Practice. In: Schuler, D., Namioka, A. (eds.) Participatory Design: Principles and Practices, pp. 239–253. Lawrence Erlbaum, Hillsdale (1993)
5. Jordan, B.: Ethnographic Workplace Studies and CSCW. In: Shapiro, D., Tauber, M., Traunmuller, R. (eds.) The Design of Computer Supported Cooperative Work and Groupware Systems, pp. 17–42. Elsevier Science, Amsterdam (1996)
6. Nardi, B.A.: A Small Matter of Programming: Perspectives on End User Computing. MIT Press, Cambridge (1993)
7. Star, L.S. (ed.): Ecologies of Knowledge: Work and Politics in Science and Technology. State University of New York Press, Albany (1995)
8. Lieberman, H. (ed.): Your Wish Is My Command: Programming by Example. Morgan Kaufmann, San Francisco (2001)

9. Lieberman, H., Paternó, F., Wulf, V. (eds.): End-User Development. Springer, Dordrecht (2006)
10. Nardi, B.A., O'Day, V.L.: Information Ecologies: Using Technology with Heart. MIT Press, Cambridge (1999)
11. Forlizzi, J.: The Product Ecology: Understanding Social Product Use and Supporting Design Culture. International Journal of Design 2(1), 11–20 (2008)
12. Jung, H., et al.: Toward a Framework for Ecologies of artifacts: How Are Digital Artifacts Interconnected within a Personal Life? In: NordiCHI 2008, pp. 201–210. ACM, New York (2008)
13. Beyer, H., Holtzblatt, K.: Contextual Design: Defining Customer-Centered Systems. Morgan Kaufmann, San Francisco (1998)
14. Beirne, M., Ramsay, H., Panteli, A.: Participating Informally: Opportunities and Dilemmas in User-Driven Design. In: PDC 1996, pp. 209–217. CPSR, Palo Alto (1996)
15. Clement, A.: Computing at Work: Empowering Action by 'Low-Level Users'. Communications of the ACM 37(1), 53–63 (1994)
16. Iivari, J., Iivari, N.: Varieties of User-Centeredness: An Analysis of Four Systems Development Methods. Information Systems Journal 21, 125–153 (in press, 2011)
17. Oudshoor, N., Pinch, T. (eds.): How Users Matter: The Co-Construction of Users and Technology. MIT Press, Cambridge (2003)
18. Kay, A.: Foreword. In: Cypher, A., et al. (eds.) Watch What I Do: Programming by Demonstration, pp. xi–xvi. MIT Press, Cambridge (1993)
19. Bødker, S., Christiansen, E.: Designing for Ephemerality and Prototypicality. In: DIS 2004, pp. 255–260. ACM, New York (2004)
20. Floyd, I.R., Twidale, M.B.: Learning Design from Emergent Co-Design: Observed Practices and Future Directions. In: PDC 2008 (2008), `http://hdl.handle.net/2142/9806`
21. Greenbaum, J.: A Design of One's Own: Toward Participatory Design in the United States. In: Schuler, D., Namioka, A. (eds.) Participatory Design: Principles and Practices, pp. 27–37. Lawrence Erlbaum, New Jersey (1993)
22. Iivari, J., Igbaria, M.: Determinants of User Participation: A Finnish Survey. Behaviour & Information Technology 16(2), 111–121 (1997)
23. Iivari, N., et al.: Mediation between Design and Use - Revisiting Five Empirical Studies. Human IT 10(2), 119–164 (2009)
24. Bødker, S., Iversen, O.S.: Starting a Professional Participatory Design Practice - Moving PD Beyond the Initial Fascination of User Involvement. In: NordiCHI 2002, pp. 11–18. ACM, New York (2002)
25. Nardi, B.A., Engeström, Y.: A Web on the Wind: The Structure of Invisible Work. Journal of Computer Supported Cooperative Work 8, 1–8 (1999)
26. Henfridsson, O.: Beyond the Container-View of Context in IS Research. In: ECIS 1998, pp. 673–683 (1998)
27. Engeström, Y.: Learning by Expanding. An Activity-Theoretical Approach to Developmental Research. Orienta-Konsultit, Helsinki (1987)
28. Kuutti, K.: Activity Theory, Transformation of Work, and Information System Design. In: Engeström, Y., Miettinen, R., Punamäki, R. (eds.) Perspectives on Activity Theory, pp. 360–375. Cambridge University Press, Cambridge (1999)
29. Engeström, Y., Ahonen, H.: On the Materiality of Social Capital: An Activity-Theoretical Exploration. In: Engeström, Y. (ed.) Activity Theory and Social Capital, pp. 1–16. Helsinki University Press, Helsinki (2001)

30. Syrjänen, A.-L., Kuutti, K.: Analysing IT and Communities of Practice. In: ECIS 2006, CD-ROM (2006)
31. Newman, M.W., et al.: Designing for Serendipity: Supporting End-User Configuration of Ubiquitous Computing Environments. In: DIS 2002, pp. 147–156. ACM, New York (2002)
32. Won, M., Stiemerling, O., Wulf, V.: Component-Based Approaches to Tailorable Systems. In: Lieberman, H., Paternó, F., Wulf, V. (eds.) End-User Development, pp. 115–141. Springer, Dordrecht (2006)
33. Huysman, M., Wulf, V. (eds.): Social Capital and the Role of Information Technology. MIT Press, Cambridge (2004)
34. Clemmensen, T.: Community Knowledge in an Emerging Online Professional Community. The Case of Sigchi.Dk. Knowledge and Process Management 12(1), 43–52 (2005)
35. Fischer, G., Giaccardi, E.: Meta-Design: A Framework for the Future of End-User Development. In: Lieberman, H., Paternó, F., Wulf, V. (eds.) End-User Development, pp. 425–457. Springer, Dordrecht (2006)
36. Engeström, Y. (ed.): Developmental Work Research: Expanding Activity Theory in Practice. Lehmanns Media, Berlin (2005)
37. Virkkunen, J., Kuutti, K.: Understanding Organizational Learning by Focusing On "Activity Systems". Accounting, Management and Information Technology 10(4), 291–319 (2000)
38. CMMI-SVC: CMMI® for Services, Improving Processes for Providing Better Services, Technical Report, Software Engineering Institute, Carnegie-Mellon University (2010)
39. Malhotra, Y.: Business Process Redesign: An Overview. IEEE Engineering Management Review 26(3) (Fall 1998)
40. Toulmin, S., Gustavsen, B. (eds.): Beyond Theory. Changing Organizations through Participation. John Benjamins Publishing Company, Amsterdam (1996)
41. Rogers, Y.: Reconfiguring the Social Scientist: Shifting from Prescription to Proactive Research. In: Bowker, G., Star, L.S., Turner, B. (eds.) Bridging the Great Divide: Social Science, Technical Systems and Cooperative Work, pp. 57–78. Lawrence Erlbaum, Hillsdale (1997)
42. Engeström, Y.: Kehittävän Työntutkimuksen Peruskäsitteitä (Basic Concepts of Developmental Work Research). Aikuiskasvatus 4, 156–164 (1985)
43. Kerosuo, H., Kajamaa, A., Engeström, Y.: Promoting Innovation and Learning through Change Laboratory: An Example from Finnish Health Care. Central European Journal of Public Policy 4(1), 110–131 (2010)
44. Engeström, Y., et al.: The Change Laboratory as a Tool for Transforming Work. Life Long Learning in Europe 2, 10–17 (1996)
45. Engeström, Y.: Putting Vygotsky to Work: The Change Laboratory as an Application of Double Stimulation. In: Daniels, H., Cole, M., Wertch, V. (eds.) The Cambridge Companion to Vygotsky, pp. 363–382. Cambridge University Press, Cambridge (2007)

Infrastructuring When You Don't – End-User Development and Organizational Infrastructure

Johan Bolmsten[1,2] and Yvonne Dittrich[1]

[1] IT University of Copenhagen, Rued Langaards Vej 7, 2300 Copenhagen S, Denmark
[2] World Maritime University, Citadellsvägen 29, 21118 Malmö, Sweden
{jb,ydi}@itu.dk

Abstract. Technologies promoting End-User Development enable domain experts to adjust and develop tools to fit with their specific work practice and thus to be efficient with respect to their professional tasks. In today's organizations, however, single applications become part of organizational infrastructures. Such infrastructures enable integration between different applications and tasks but, at the same time, introduce constraints to ensure interoperability. How can the advantages of End-User Development be kept without jeopardizing the integration between different applications? The article presents an empirical study on End-User Development in the context of the development of an organizational IT infrastructure. Based on the analysis of the empirical material we discuss the challenges the infrastructure context provides for End-User Development.

Keywords: End-User Development, IT Infrastructure Development.

1 Introduction

End user development (EUD) allows users to develop and evolve their computer based working tools to support their specific tasks in an efficient way thus enabling users to be more effective. EUD possibilities furthermore allow innovations of processes and work practices to be mapped easily into the supportive technology.

In today's organizations, single applications become more and more part of joint technical infrastructure supporting the cross-organizational and sometimes inter-organizational cooperation. The necessary standardization can be expected to constrain the freedom for specific adaptation and development on a local level. [1]

The majority of the contribution to the End-User Development (EUD) discourse focuses on the tailoring and development of specific applications – like spreadsheet systems [2], or CAD systems [3] – or individual parts of infrastructures – like search tools [4]. Few contributions address the appropriation of common communication and cooperation infrastructures. (See [5] as an exception.)

Based on an empirical study of End-User Development the article explores two related questions: What are the challenges End-User Developers face when developing (parts of) an IT infrastructure? How do they tackle them? What can be done to keep the advantage of EUD and when integrating applications to an IT infrastructure?

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 139–154, 2011.

The study is set at the World Maritime University (WMU) in Malmö, Sweden.. Under the auspices of the of the International Maritime Organization (U.N), WMU is an international university that provide Master degree educations in the maritime area for around 250 students predominately from developing countries each year.

The research presented here is part of a more comprehensive study on Participatory Design (PD) as basis for infrastructure development in an intercultural organization. During the research, WMU has been moving towards a more integrated technical and organizational infrastructure to consolidate the university's IT systems. EUD has been part of the organizational ICT development practice from the very beginning. As the innovative potential of EUD and the contribution of domain experts to the design of common infrastructure has early been recognized, the subject of this article is an important contribution to the development of an organizational IT strategy.

The cases subject to this article have been selected because the End User developers have both been active for more than 20 years. Their development activities have been acknowledged as important for the organization. As the scope, technical sophistication, and size and character of the user community differ significantly, the cases together provide a consolidated picture of EUD at WMU.

The remainder of the article is structured as follows: Section 2 summarizes the relevant literature on organizational End-User Development and infrastructure development in order to provide a conceptual framework for the article. Thereafter we introduce the research methods. The empirical section presents the two cases. In the following, the analysis is presented and discussed. The challenges of EUD in infrastructure settings are developed with respect to five aspects. The central and main dimension is the fragility of EUD practices in an organizational context. This fragility also influences cooperation with developers and users. The informal character of EUD practices makes it difficult to coordinate with professional development. And finally the technical platform and its development provide a challenge for EU developers.

2 End User Development, Organizations, and Infrastructures

Early research in EUD mainly focused on development and tailoring of individual performance tools in single user work environments like e.g. computer aided design [3, 6], excel sheets [3]. However, already then the cooperation between End User Developers (EU developers) and other users became visible as an important theme. [3] Organizational support for the development and maintenance of common customizations has been addressed as well. Only a few articles address EUD in the context of infrastructure development. Empirical research on infrastructure development, though, indicates the importance of bottom up, participatory approaches to keep the infrastructure in line with developing organizational requirements.

2.1 Tailoring of Common Tools

Already one of the first articles on tailoring [7] reports on EU-Developers exchanging self developed features and add-ons with other users. As early as 1992, Gantt and Nardi describe patterns of cooperation between EU-Developers and other users of

CAD systems. [8] They observe the development of formal and semiformal positions in organizations where local developers do not only act as 'gurus' – acquiring and sharing knowledge about how to tweak the system on an individual base – but as 'gardeners' – maintaining a set of customizations and tailorings for their group or department and continuously enhancing the common work tools and thus improving the productivity of the whole team.

Early on, support for sharing and cooperation among user communities has been a research topic. (see e.g. [9]. Pipek provides in [10] a categorization of cooperative tailoring scenarios: *Shared usage* requires the least coordination and user groups are a self help feature in both commercial and private contexts. Cooperative tailoring in a *shared context* provides better possibilities for sharing customizations, but might result in conflicts if changes to the individual tool hinder the sharing of work results. When users tailor a *shared tool*, they need to negotiate not only the adaptations but also the usage of the common tool. *Shared infrastructure* scenarios are least researched and provide additional challenges. Here tailoring results can effect configurations of other systems. The design space for EUD of an individual application is constrained by the interoperability requirements. Heterogeneous user groups are dependent on each other though they neither share a common work practice nor a common tool. Dittrich and Lindeberg discuss such a case; in infrastructures that support data-intensive businesses like telecommunication, the flexibility of a specific application can only be deployed when other applications in the same network and the interoperability platform can be tailored accordingly. [11] The importance of combining EUD and professional development activities when evolving such a common infrastructure and support for it is addressed in. [12]

2.2 Organizational Support for EUD

Already Gantt and Nardi [8], [3] emphasize the importance of organizational recognition of End-User Developers. Other researchers emphasized the contribution of 'gardener type' local designers as well. The notion of shop floor IT Management [16] highlights the importance the users' work that makes IT-infrastructures work. Dittrich et al. report about local designers developing and maintaining infrastructures for municipal service provision. [13] Kanstrup presents a study on a local IT-support by a gardener type EU developers and in depth analyzes the practices that allow them to foster the IT-use in their organizations. [14] However, EUD done "on the behalf of the organization or group in which they [the EU developers] work" [15] needs to be deliberated with this group. Trigg and Bodker discuss this phenomenon as systematization of EUD. [15] Beyond addressing the EU developer's requirements, organizational EUD might require technical, organizational and in Trigg and Bødker's case even legal deliberation. [15, 17] The organizational forms and methods to do so are so far only discussed as challenges. [17]

2.3 Infrastructure Development

Infrastructure development is discussed in the discourses on e-Governance and Enterprise Architecture. The mainstream of these discourses emphasizes a control oriented perspective on IT infrastructure development. (See for example [18, 19].) Empirical research however challenges the feasibility of a rigor top-down approach. [20] Furthermore, to support innovation and creativity, the IT infrastructure of an organization needs to be flexible and accommodate bottom up design-in-use. [21] Karasti and her co-authors emphasize the need for PD and shop floor IT-management in the context of scientific infrastructure development and evolution. [22, 23] One of the core aims of the research project the current study is part of is to understand and develop PD as an organizational implementation strategy. [24]

Up until now, little research addresses tailoring in the context of IT-infrastructure evolution. Wulf et al. develop and investigate the usage and tailoring of a fully flexible search tool. That tool was part of an infrastructure supporting distance collaboration between a Bonn and Berlin office when the German government moved from one city to the other. [4] Studies of continuous infrastructure tailoring would have been interesting in this respect. However, the usage of the platform was not continued when the research project ended. [25] Eriksson emphasizes the need to relate tailoring and professional software engineering when evolving infrastructures for data intensive businesses like telecommunications [26]. In earlier research, we reflected of how the technology used for the implementation of infrastructure influences the possibility for user participation in design and the space for tailoring. [27]

The study presented here addresses practices of organizationally recognized EUD before, during, and after the introduction of an integrating infrastructure to. It allows addressing the complex interaction between EUD practices and IT infrastructure evolution.

3 Research Methods

The research project this study is part of follows Cooperative Method Development (CMD) [28] as a research approach. CMD anchors process and method improvements in the understanding of shop floor development practices. Empirical research aiming at understanding the practitioner's problems is followed by a joint deliberation of improvements and the implementation of the improvements that again are accompanied by empirical research. This study positions itself as a case study in the first phase of a new cycle where the idea is to further the understanding of the EUD activities in the context of infrastructure development. This enables more informed deliberations of improvements later.

The first author combines his PhD studies with his work as faculty IT specialist at WMU. The second author is the supervisor of the PhD project and has also participated in the data collection.

Data gathering and analysis. Though the cases have been selected based on previous research, the empirical material analyzed here has been collected to understand EUD practices from the EU developers' point of view. The empirical material entails both participatory observation and interviews.

With respect to the first case, Liz developing electronic forms and a contact database, the first author carried out two participatory observation sessions. The sessions were conducted in Liz's office in front of Liz's computer. Liz practically showed how she worked with the electronic forms and the contact database. At the same time, she exemplified her approach regarding the design and implementation process. For this purpose sketches and paper printouts of relevant artifacts were also used. The second author, as an external actor and research supervisor, later conducted a follow-up interview. The purpose of the interview was both to relate back to findings of the participatory observation sessions and to inquire about Liz's relation to official IT-development beyond the specific EUD activities. Her experience of how the organization of IT development has changed over time provided valuable insight into the impact of organizational IT development on EUD practices. All material gathered has been recorded, transcribed undergone qualitative analysis.

With respect to the second case, John's development of WMU's registry system, six participatory observation sessions have been conducted with the Registrar John. These were done explicitly in the context of an upcoming ERP project where the intention was to understand if and how the current registry system could be integrated in the new application environment. All sessions were carried out in John office according to the same manner as the former case. An interview with the senior registry assistant, Sue, was conducted at an offsite location to get an alternate perspective. Three of the participatory observation sessions together with the follow-up interview have been transcribed and undergone qualitative analysis (all material is recorded).

The transcriptions have been analyzed with the qualitative research tool HyperResearch. The analysis started with identifying codes in the transcribed material. Based on this open coding, a number of categories where developed which where used for axial coding, relating the different transcripts. The categories provide the structure for the analysis below.

Trustworthiness. To assure the validity of the research, we applied various triangulation strategies. Already the two cases allows for cross case triangulation. Within each of the cases we used different data collection methods: participatory observation and interviews. Where we judged that the main researcher was too involved in the organization, the interview was implemented by second author who is not been part of the organization. In the registry case, a second person, a representative of the user group of the registry system was interviewed as well. The empirical research presented here is triangulated by earlier and parallel long term ethnographical research. Where relevant we draw on this additional research in the presentation of the analysis results. The rich description provided below provides the reader with the means to judge the conclusion drawn based on the analysis. The final as well as intermediary analysis results have been checked with the EUD practitioners whose practice was subject to the research presented here. However, as qualitative research and based on a single organization, we do not claim generalizability of the results. It is left to future research to confirm, contest and detail the findings of this article.

4 The Two Cases

4.1 Case 1: Electronic Forms and Contact Database

The End-User developer of our first case is Liz, a long-term and today senior administrative assistant at WMU. Liz has been a member of staff for almost thirty years and has been part of the university's journey from a manual typewriter operation to an increasingly integrated technical infrastructure. Talking with her, one recognizes her genuine interest in smart solutions, which save time and effort. E.g. when she refers to her first encounter with computer based forms: "So I learned that you can do online forms […]. I thought this was just the best thing since sliced bread."

Her role in developing IT support for administrative purposes for the whole organization is acknowledged, but not organizationally defined in for example her work description. Referring to this semiformal position, Liz describes herself as "sort of a spider in the net". For eight years she was also a member of the computer committee that gathers key domain experts and IT developers deciding on the IT infrastructure, until WMU hired professional IT developers to work with IT-support for faculty and related administrative tasks.

Two of Liz's areas of responsibility are to administrate (1) internal forms such as leave and travel requests and (2) a repository of WMU of contacts. From the beginning, these were based on paper and typewriter. This started to change when word processing programs with contemporary features became available. Liz especially recalls version eight of Word Perfect where it became possible to set up electronic forms:

> *"We are going back 20 years you know. […] I thought it was super […] So I use help a lot, and I have learned to read the screen […] I went through it step by step you know. Click on the name, textbox and fields, and all that you know. I learned about the fields. Trial and error, first it didn't work you know. So I made a leave request form […] I take a form that is for everybody, then you get something that is across the board. And my boss at that time […] I tried it on him of-course".*

Liz ended up not only migrating the leave request form to Word Perfect, but also the rest of the administrative forms.

In addition, contact information also began to be maintained in Word Perfect. Many administrators experienced the initial approach as insufficient: Contact information became scattered throughout the organization. In order to get hold of information about a certain person, one had to know who internally was maintaining a particular record. This led to a discussion of the benefits of having a central point of reference for contacts: a database with generic and standardized fields appropriate for different functions that anybody could access and query. In the end, the development of a Microsoft Access database was decided on. All the administrative assistants and secretaries were sent on a Microsoft Access short course to be able to develop and maintain the database internally. Upon their training, Liz - then already known for her technical interest and expertise - ended up taking charge of the development of the contact database, creating both the database and associated interfaces. The idea was that the secretaries would primarily be in charge of inputting data, whereas professors and others also could extract it. In the end, the contact database contained altogether about contact 640 records.

For both the electronic forms and the contact database, Liz gradually developed a model for user involvement. In regard to the electronic forms Liz for example describes how she works with actively getting feedback from other users. Acknowledging a wide range of competences, she has developed an implicit ranking of users from computer illiterate to technical experts that she tests prototypes on. Already when migrating the forms to Word Perfect, Liz started to work with different colors, fonts, and layouts to make the user experience more intuitive for the different user groups and purposes. In regard to the contact database, she produced manuals and trained the other secretaries of how to use the interfaces of the database.

Both the electronic forms and the contact database have undergone major revisions. For the electronic forms, the next major technical infrastructure change was an organization wide change to the Microsoft Office suite and Word. For Liz this meant that it was back to the books and the help files to learn. The Word version of the forms were in operation for 18 years and became the de facto standard in the organization and also part of other technical infrastructures such as the web-based intranet. However, although new and advanced technical features became available with the Word-based forms such as mail merge and calculation capabilities, Liz was never altogether satisfied with the format. She experienced Microsoft as more "fuzzy […] if you are a new user to forms". The latest revision embarked on involves using pdf and adobe life cycle as a technical base. This change enables a full integration of the forms with other applications, and allows Liz to continue to improve usability aspects. For the technical integration, Liz has had to learn how the XML based backend of the forms works. She has then been involved in creating several prototypes in cooperation with one of the IT-program officers, where the forms exchange information via web-services with the in-house intranet. In addition, the easier design of the pdf forms has opened up for Liz to train other EU developers to create their own forms.

The biggest changes to the contact database have been of organizational nature. Coming up to four years in operation, the contact database and Liz's role in developing and maintaining it became subject to fluctuation. One after another, the other administrators retired or left the organization. At the same time, the university started to build a professional IT department. This for example meant that Liz was relieved from the coordinating computer committee and replaced by IT professionals hired to support the faculty and related administration. Gradually, this implied a disruption in the organizational anchoring of the usage and development of the contact database. The Access database was turned into a dedicated address database with Liz herself as also the main user. During the last years, again a discussion has emerged about the benefits of having a central contact database. The first pursuit to re-establish such a database, came with the development of a new external website. Using the contact database as a foundation, Liz again became involved in implementing new features and updating the contact records. The intention is that these records subsequently will be moved to an ERP system in the pipeline for implementation.

4.4 Case 2: The Registry System

The registry system for WMU was developed from scratch by its Registrar John. John came to WMU from the United States in 1992 and had already then comprehensive experience of the function from American universities. In the United States, technical expertise is often a mandatory requirement for the Registrar. Being able to operate databases and reporting tools to for example extract student data to provide decision support is a fundamental task. Today, the most common off-the-shelf system is Banner. When John initiated his career, such systems were still in a pilot stage. Before John started his employment at WMU, he participated as a domain expert in the development of an early registry software tool. Through this involvement, he advanced his technical expertise by gaining his first experience of high-level programming.

The registry function at WMU is modeled partly after the US system, which implies that the Registrar holds a managerial position on the same level as a Vice-President. The registry department at WMU is made up of four employees: the Registrar John, the Associate Registrar, the Student Services Officer, and the Senior Registry Assistant Sue.

In 1992, WMU did not have satisfactory university standards for core registry functions such as course, subject, credit, and grade management. Instead of subjects that had a direct relation to weighted credits, programs were made up of modules that defined broad teaching areas. The modules were not individually graded, and the certificates presented to the graduates only contained an overall evaluative statement "Can you imagine, can you imagine, coming into this situation?" John reflects back.

So when John initiated his employment at WMU, he gradually started to construct an accountable academic management system. Parallel, he began to design and implement a computerized system himself. Building on his previous experiences, he picked a high-level programming language and database called DataFlex together with the reporting tool Crystal Reports. After a number of generations of the system – alone the grade management has undergone eight successively evolutions – John believes that he has succeeded towards accomplishing his initial vision: "what we do is we built the system to basically do our jobs, all of our jobs, all the four people in the registry, and that was exactly the purpose, to go from a manual paper based operation, to a computerized electronic method". Today, a dedicated and tailored computer support is in place for major WMU registry functions such as admissions, student profiles, courses and subjects, grade management, and quality assurance. The system has come to contain additional components such as alumni records and hostel management.

Despite the vision of a comprehensive computerized system, some processes still need a combination of electronic and paper operations. The start of the admission process is for example marked with the registry department receiving a paper application form. The data is transferred into the registry system. Thereafter, the application is subject to a complex admission process involving both internal committees and external agencies, which is supported and documented in the system. Once the student is admitted, all study activities and results are documented as well

Sue has been one of the main users of the system cooperating with John around the design of it. She recalls the evolution of the data entry interface: "I know that in the beginning, when I started, these tabs where divided in three different databases, and I thought it was rather complicated to remember which tabs that belonged to which [...] you can always call him, go in to him, and he listens [...] it is not like it is a small petites, he does do, writes it down on his little notepad. I have not thought about it before, but now when we are talking about it, it is pretty great [...] and then he either says it works, if it works [...] when he says it doesn't, it is because it must be possible to extract some report".

Members of staff outside the registry department, though, are less satisfied with their access to the registry system: they cannot for example extract reports from registry system apart from a number of pre-defined template reports without acquiring substantial knowledge about the database structure and the report tool. According to

John, he down prioritized requirements from outside the registry department due to time constrains: "the whole concept behind this wasn't to be for the university, it was supposed to be for the registry only, and then we decided to give it to people, it wasn't meant to be the ERP system for the whole university, it was for us to get our work done, and then people wanted things so, I then, I had to go in there, and then they were never happy, the main thing is I would have had a full time job just coding this".

5 Analysis and Discussion

In this section, we both analyze the field material and discuss the findings. The subsection headings were derived from the field material in the manner described in the method section. In each subsection we start with introducing the theme we then summarize and cite from our empirical research and finally discuss the implications for supporting EUD in the context of infrastructure development and evolution.

5.1 Maneuvering as an Informal Developer

As an informal or semi-formal developer, EU developers are in a vulnerable position. On the one hand, they develop part of the IT infrastructure for the whole organization and provide important tools. They are aware of their role, and e.g. consciously include relevant stakeholders. On the other hand, as the episode with contact database shows, they are not officially recognized as developers and other organizational actors might not be aware of their activity, especially when personnel changes.

Liz: An unprofessionally professional developer. Though also being one of the main beneficiaries of her work, Liz consciously targets other staff with her development. She not only gathers 'requirements' in an informal way but consciously addresses lifecycle management such as training, further development, and maintenance. She for example does not only develop the electronic forms, but runs informal user tests and provides help. The contact database comes with a user manual. Both are maintained and adjusted to changing requirements and technologies. While her EU-development in many ways resembles that of an IT-professional, she is only informally recognized as a champion user in the organization. IT-development is nowhere to be found in her work description. Acquiring new IT competences is often done on her spare time and she for example carries expenses for books. Though her efforts are appreciated, her ability to maneuver in the organization – for good and for bad – is affected by her "unprofessional" status. E.g. where her IT-professional colleagues should take part in IT- forums such as the computer committee as part of their role, this is not the case for Liz.

The necessary cross-departmental IT-coordination takes place in a different way: Liz describes herself as "sort of a spider in the net" when it comes to development and coordination of the IT related ventures she has been involved in. This allowed her to continue being part of the forms development and the contact database, just that the coordination takes place outside the formally arranged forums. In the latest attempt to revive the contact database in the context of a new external website, it was WMU's president that turned to Liz to assist with the coordination. The reason is that through her day-to-day work, she has an established relationship to staff stakeholders and knows who to ask for requirements and how different people could contribute. Liz has continued to maintain her relationships with the professional IT developers and in many cases acts as a broker between the users and the IT developers.

John: "The captain that controls all the pieces" Like Liz, John is not explicitly recognized as an IT professional at WMU. However, as a Registrar, he is a senior management member. With respect to his development mandate, this implies that John has space to basically carry out development for the registry system as he sees fit as long as his department meets the university's overall expectations. To this end, John has also taken on tasks beyond the 'normal' EUD and developed support for the whole registry and integrated his registry system with e.g. the office suite. Even though he also is a central beneficiary himself – throughout the interviews both John and Sue emphasize that the core function the of registry system is to output student management reports – the client interfaces and integration with other software such as mail merge for Microsoft Office are on the grand scheme more used by the other registry staff.

In his capacity as a manager and (recognized) key domain expert of his department, John has a permanent place in the computer committee. However, except from securing his annual development budget, his use out of the committee is limited. John's vision was to build an "electronic method" for all core functions of the registry department, but only for the staff members of the registry department. At the same time as the registry department has the most comprehensive support, all input into and export of information beyond the department is done manually. E.g. grades arrive to the registry department in an electronic format, but have to be manually transferred one by one. Though some client interfaces for faculty exist, people call John and ask for different reports to be exported.

During the interviews, the possibility to integrate the registry system with the surrounding infrastructure was discussed. It turned out that this would have been technically possible. However, the protective attitude of John that enabled him to develop a comprehensive and consistent application, hindered an earlier exploration of such possibilities.

Discussion. EU Developers have a vulnerable position in the organization, as their expertise both regarding their development tasks and regarding the organizational needs are often not recognized. Liz and John follow different strategies to cope with this challenge. Liz follows a networking approach whereas John uses his role as a manager to define the borders and control use and development of the registry system. When establishing an organization to coordinate infrastructure development their 'shop floor IT-management' [16] and their ability to act as brokers between users and IT professionals [14] need to be preserved for the benefit of the organization. Adequate forms of representation need to be established.

5.2 Frontline User Cooperation

When EU Developers develop systems for others to use, cooperation with these other users is important as well. Not surprisingly both the EU Developers we interviewed and observed have an established practice to involve other users in the development.

Liz: An EU Developer learning about usage. Rightfully, Liz describes herself as "one of them" – her users – and claim that she has a good conception of how the contact database and the electronic forms will be used. However, instead of only using herself as a reference user, she also works actively to understand the perspectives of other users through for example prototyping and testing against different stakeholder segments. The reason is that she is directly confronted with the problems other users have with her applications. As an administrative assistant she is placed in the middle of organizational activities with long-term established relationship with other staff. She is, therefore, one of the first to get notified if her development ventures do not work: "I end up with more questions then, and if there 's is more question I end up with people who don't use it." And people "who don't use it" mean more work for herself.

John: Caring for his users. The motivation for John to involve his department in the development is a different one. The change, e.g., initiated by Sue was not a malfunction per se. The program was fully functional. Its prior design was developed in accordance to the preferences of the previous registry assistant. The changes that Sue called for involved John changing both the interfaces and the database.

During the interview with Sue, she compared the way John cooperates with the members of the registry department to previous experiences. She worked as a secretary at a major company during a migration to SAP: "I mean, there was never any question of us having any input to it. It was like it was, but they had some sort of groups, from different department where they went through what was needed. But then afterwards, it was like it was. […] But I guess, there are pros and cons with everything". One con is raised against John's way of development: "Honestly speaking, it can appear a bit stiff, for example you have to save here, there. One perceives it as a bit old fashioned one enters information."

Discussion. EU developers care about usability; they are confronted with the problems of not usable software; and they develop ways to cooperate with the users of their systems around their design and development. The EU developers' expertise could be used by professional developers when working with IT infrastructures: As members of the user community and as shop floor IT managers, they might be able to help with recruiting the right people for user participation and also be able to prioritize between crucial problems leading to users refusing an application and 'good to have' features that can wait until developers have time.

5.3 From One Software Developer to Another

Modern IT infrastructures for educational organizations with needs to support both external and internal cooperation are not possible to maintain without professional IT developers. WMU decided to have IT competences close to faculty and administration. Over the last 7 years, two fulltime positions were established. This requires the EU developers to cooperate with their IT professional colleagues.

Liz: Including the professional developers in her network. IT professionals are colleagues too. Liz's way of managing the professional IT developers is to include them in her network as well. That way she is consulted and included in the development interfacing with and impacting her applications. In the case of the pdf forms she for example negotiated the backend development with the IT-professionals in order for the results to be compatible with a future integration in a wider IT-infrastructure. In regard to the updated contact database she had to coordinate the interface development with both professional developers and contributing staff members. As the integration into the infrastructure poses new technical problems this cooperation includes opportunities for learning new technologies.

John: Isolating the own application. Due to the need to limit the complexity of an already complex system, John isolated the own application from the development of IT-infrastructures around the registry department. One result of this is that the possibility of technical integration has not been explored.

Discussion. In the context of infrastructure development, the cooperation between professional IT developers and EU developers seems to be a more viable strategy in order to coordinate more substantial technical development with the EUD of parts of the infrastructure. E.g. evolution of the electronic forms and the contact database needs to be coordinated with the Infrastructure development. The formal organization of the IT infrastructure development needs to accommodate the need for coordination and cooperation between professional and EU development.

5.4 … Something that Otherwise Would Be Defined as a Project

The need to coordinate EUD and professional development of the same infrastructure has been highlighted above. However, already the ‘gardening’ metaphor coined based on previous research [7, 14] indicates that EUD takes often place without a formal project organization, interlaced with the actual tasks of the EU developer.

Liz: Focusing on specific applications. Both the development of the electronic forms and contact database would normally be defined as projects, except that in Liz‘s case they don’t qualify as such. At least not according to what can be recognized traditional IT-project criteria like predefined scope, resources, and start and end point. The scope is negotiated between Liz and her users; the time resources are found whenever there is no urgent other task; the whole ends when there is nobody using the results anymore. There is no project charter, no formally defined objective, identified constraints and stakeholders. Even the implementation platform changes over time. However, both development activities are clearly limited. The forms development is about administrative forms. Requests to develop forms for other departments are answered by teaching the person to do it him or herself. The contact database is about people and addresses. Other functionality did vary over time.

John: Developing for the registry department. Also John did not organize even major revisions of the registry system in any formal way. When Sue is asked about how improvement proposals are handled, she answers: “Ehh, I don’t know, I was just happy that it was reduced [in reference to the databases connecting to the tabs and fields].” The developments of the registry system and the cooperation between the four staff are not done with a formal project management or a project charter. Judging from the interviews, the question however is if such measures not only would have been bureaucratic red tape. The development seems to be coordinated by informal meetings. However, John clearly limits his development activities to the support for the registry department.

Discussion. As EUD does not take place in the form of projects, it is less visible in the organizational context. Professional development needs to develop ways to coordinate infrastructure development with the more informal ways EUD takes place. Formal committees like the computer committee provide a place where some of the coordination can take place. However, it is not sure whether simply providing a meeting place scales when the organization grows beyond a size where professional and EU developers can sit around one table.

5.5 Technical Platform

From the EU developer's perspectives the technical platform provides challenges to cope with. From an organizational IT infrastructure perspective it both enables and constrains development. Below we again summarize relevant parts of the field material and discuss the implications.

Liz: combining reading manuals and trial and error. How do EU developers acquire the necessary competences for developing stabile applications for usage by others? Liz applies two strategies: Continuous trial and error and step-by-step development is used to solve technical problems arising from her everyday work. In the end, this leads to the intimate knowledge of the workings of a technical platform: "I know how they were thinking when they made it [MS Word]". The other strategy is to acquire more abstract knowledge. Liz herself emphasize the importance of her own diligence and when it comes to always reading books and manuals to learn about the existence and properties of technical features and possibilities before and during her development. She for example describes how she reads up on the workings of a new feature, makes a small prototype for testing at home, and then transforms that into working functionality in relation to a current task.

Liz's ability to assimilate de-contextualized technical knowledge and put it into practice allowed her to port the electronic forms across three different technical platforms: from Word Perfect over MS Word to the adobe suite. The last change provided an additional dimension. As the forms now should interface to databases and other applications, the data model behind the forms needed to be more independent from the form as the user sees it. To cope with this challenge, it was necessary to understand notions like data structure and mark-up languages (XML). The cooperation with one of the professional developers helped to master this learning step.

John: Technical proficiency as part of the job description. That John is a technical domain expert is not so strange: Technical proficiency is "the first thing they list in any job advertisement" in the United States. As presented above, John has acquired programming expertise. John himself developed the majority of registry system's code. In addition, DataFlex has an active community that contributes with script that has been incorporated to for example create more advanced menu structures.

Discussion. The implementation platform, respectively exchanging it, shows in our empirical material as technical challenges. Interfacing EUD results to an infrastructure does contribute additionally to the requirements for technical and conceptual know-how. From an infrastructuring point of view, the implementation platform for the infrastructure has to be selected carefully to provide the possibility to interface to heterogeneous applications and to allow for non IT professionals to use it for a base for EUD. (See [27] for further discussion.) And interfaces between EUD results and the infrastructure indicates where coordination between professional and EU developers is needed. When evolving and introducing new technical platforms, however, impacts on the EUD results need to be considered and the EU developers need to be provided the necessary support to update their technical proficiency.

6 Conclusions

The article addressed the relationship between EUD and organizational infrastructure development and evolution. EU developers' expertise in shop floor IT management and their established role as brokers between professional developers and users can provide a resource for professional infrastructure development as well. The meeting of these two different practices of development is, however, not an easy one.

In the discussion we outlined five sets of challenges. The first and central is the fragility of EUD practices due to their informal character. The cooperation with users, with professional IT developers and the scoping of EUD is subject to individual strategies. Finally, the technical platform connecting heterogeneous applications provides a challenge for EU developers, who might need help to conquer new technology.

In the two cases, two different strategies to cope with these challenges became visible: networking and isolation and control. With the growing requirement to integrate individual applications into an IT infrastructure, the former appears to be the more viable.

We identified a number of means to address the challenges from an IT infrastructure perspective: Representing EUD in organizational IT committees, fostering cooperation between EU developers and professional developers, acknowledging EU Developers on an organizational level, and selecting the technical platform and designing the infrastructure to accommodate local EUD.

Based on the analysis and discussion, we conclude that EUD and organizational infrastructure development can be combined. In our future research, we will explore how to solve the issues identified in the previous section when introducing an ERP system of WMU to integrate the financial and personnel administration with other parts of the infrastructure.

Acknowledgements

Thanks to the World Maritime University and it's End User Developers for the possibility for the research presented here; to the anonymous reviewers for their feedback.

References

1. Hanseth, O., Monteiro, E., Hatling, M.: Developing information infrastructure: The tension between standardization and flexibility. Science, Technology & Human Values 407 (1996)
2. Burnett, M., Rothermel, G., Cook, C.: An Integrated Software Engineering Approach for End-User Programmers. In: Lieberman, H., Paternó, F., Wulf, V. (eds.) End-User Development. Springer, Heidelberg (2006)
3. Nardi, B.A.: A small matter of programming: perspectives on end user computing. MIT Press, Cambridge (1993)
4. Wulf, V.: "Let's see your search-tool!"—collaborative use of tailored artifacts in groupware. In: Proc. of the Int. Conf. on Supporting Group Work, pp. 50–59. ACM, New York (1999)
5. Pipek, V.: From tailoring to appropriation support: Negotiating groupware usage. PhD Thesis, University of Oulo, Finnland (2005)
6. Mørch, A.: Three Levels of End-User Tailoring: Customisation, Integration, and Extension. In: Proc. of Computers in Context, Aarhus, Denmark, pp. 157–166 (1995)
7. Henderson, A., Kyng, M.: There's no place like home: Continuing Design in Use. In: Greenbaum, J., Kyng, M. (eds.) Design at Work. L. Erlbaum Associates Inc., Hillsdale (1992)
8. Gantt, M., Nardi, B.A.: Gardeners and gurus: patterns of cooperation among CAD users. In: CHI 1992 Proc. of the SIGCHI Conference, pp. 107–117. ACM, New York (1992)
9. Kahler, H.: From Taylorism to Tailorability. In: Proceedings of the HCI 1995, vol. 20B, pp. 995–1000. Elsevier, Amsterdam (1995)
10. Lieberman, H., Paternó, F., Wulf, V.: End-user development: An emerging paradigm. In: Lieberman, H., Paternó, F., Wulf, V. (eds.) End-User Development, pp. 1–8. Springer, Heidelberg (2006)
11. Dittrich, Y., Lindeberg, O.: Designing for changing work and business practices. In: Patel, N. (ed.) Adaptive evolutionary Information Systems. Ilea Publishing (2003)
12. Eriksson, J., Dittrich, Y.: Combining Tailoring and Evolutionary Software Development for Rapidly Changing Business Systems. J. of Org. and End-User Comp. 19, 47–64 (2007)
13. Dittrich, Y., Eriksén, S., Hansson, C.: PD in the Wild; Evolving practices of Design in Use. In: Proc. of the Participatory Design Conference (2002)
14. Kanstrup, A.M.: Local Design: an inquiry into workpractices of local it-supporters. PhD-thesis. Department of Communications, Aalborg University, Denmark (2005)
15. Trigg, R., Bødker, S.: From implementation to design: tailoring and the emergence of systematization in CSCW. In: Proc. of the 1994 ACM Conf. on CSCW, pp. 45–54 (1994)
16. Eriksén, S.: Knowing and the art of IT management: an inquiry into work practices in one-stop shops. PhD Thesis. Lund Technical university (1998)
17. Dittrich, Y., Lindeberg, O., Lundberg, L.: End-User Development as Adaptive Maintenance. Two Cases. In: Lieberman, H., Paternó, F., Wulf, V. (eds.) End-User Development. Springer, Heidelberg (2006)
18. Bernard, S.B.: An introduction to enterprise architecture. Authorhouse (2004)
19. Zachman, J.: A framework for information systems architecture. IBM Systems Journal, 454–470 (1999)
20. Hanseth, O., Braa, K.: Hunting for the treasure at the end of the rainbow: standardizing corporate IT infrastructure. Computer Supported Cooperative Work 10, 261–292 (2001)
21. Ciborra, C., Andreu, R.: Organizational learning and core capabilities development: the role of IT. Wiley Information Systems Series, pp. 87–106 (1998)

22. Karasti, H., Syrjänen, A.: Artful infrastructuring in two cases of community PD. In: Proc. of the 8th Conference on Participatory Design, pp. 20–30. ACM, New York (2004)
23. Karasti, H., Baker, K.S., Millerand, F.: Infrastructuring for the Long-Term: Ecological Information Management. Comput. Supported Coop. Work 19, 377–415 (2010)
24. Bolmsten, J., Dittrich, Y.: Who's Afraid of the Big Bad Wolf? - Leveraging the Capabilities of Shop Floor IT Design on the Organizational IT Management Arena (work in progress)
25. Pipek, V., Wulf, V.: A groupware's life. In: Proc. of the 6th ECSCW, pp. 199–218. Kluwer Academic Publishers, Norwell (1999)
26. Eriksson, J.: Bridging the Gap between Development and Use - Support of Tailorability in Software Evolution. Lic. Thesis. Blekinge Inst. of Techn., Ronneby, Sweden (2005)
27. Bolmsten, J., Dittrich, Y.: Technology Matters - The role of the technical base in infrastructure development and evolution (work in progress)
28. Dittrich, Y., Rönkkö, K., Ericksson, J., Hansson, C., Lindeberg, O.: Cooperative method development. Empirical Softw. Engg. 13, 231–260 (2008)

Semiotic Traces of Computational Thinking Acquisition

Clarisse Sieckenius de Souza[1], Ana Cristina Bicharra Garcia[2], Cleyton Slaviero[2],
Higor Pinto[2], and Alexander Repenning[3]

[1] Departamento de Informática, PUC-Rio
Rua Marquês de São Vicente 225, Rio de Janeiro – RJ, Brazil
clarisse@inf.puc-rio.br
[2] Instituto de Computação, UFF
Rua Passo da Pátria 156, Niterói – RJ, Brazil
bicharra@ic.uff.br, cslaviero@gmail.com, higor@bcc.ic.uff.br
[3] Computer Science Department, University of Colorado at Boulder
430 UCB Boulder, CO 80309-0430 - USA
ralex@cs.colorado.edu

Abstract. Computational thinking involves many different abilities, including being able to represent real and imaginary worlds in highly constrained computer languages. These typically support very selective kinds of perspectives, abstractions and articulation compared to the unlimited possibilities provided by natural languages. This paper reports findings from a qualitative empirical study with novice programmers, carried out with AgentSheets in a Brazilian public school. The driving research question was: How do meanings expressed in natural language narratives relate to computational constructs expressed in programs produced by novices? We used semiotic and linguistic analysis to compare meaning representations in natural and artificial texts (game descriptions in Brazilian Portuguese and Visual AgenTalk code). We looked for recurring relations and what they might mean in the context of computational thinking education. Our findings suggest that the semiotic richness of AgentSheets can be explored to introduce different aspects of computational thinking in principled and theoretically-informed ways.

Keywords: Computational thinking education; End user programming languages; Semiotic analysis; Discourse analysis; AgentSheets.

1 Introduction

A prime requirement for end user development to succeed is that end users be able to think computationally. Among other things, end users must have the ability to express *what they mean* in computable form, that is, to build representations of real (or imaginary) objects and phenomena using constructs of highly constrained computer languages [12]. These typically support only very selective kinds of perspectives, abstractions and articulation compared to natural languages, which support the expression of virtually anything that speakers can conceive of.

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 155–170, 2011.

For novices, incremental elaboration of computer representations usually starts from imprecise mental representations that can be expressed in equally imprecise natural language discourse. When this sort of discourse is externalized, it creates tangible instances of signs that support subsequent semiotic transformations until formal and precise expressions of meaning can be used to compose computable code fragments. These fragments of artificial code blend together with natural signs and expand the semiotic universe of the novice programmers, helping them to compose larger structures of representations which eventually constitute a meaningful and executable computer program.

In this paper we explore the connection between natural language representations of program meanings and their corresponding computational encoding. We discuss findings from a qualitative empirical study with novice programmers, carried out during a Scalable Game Design project [18] with AgentSheets [16] in a Brazilian public school. We used semiotic and linguistic analysis to compare meaning representations in game descriptions expressed in Brazilian Portuguese and their corresponding program version in AgentSheets. We looked for recurring relations in the data collected from a group of 9th graders that had no previous training in computer programming. Our goal was to find what these relations might mean in the context of computational thinking education. As is the case with all qualitative research, our findings are not predictions about novice behavior. They point to three factors that may play a role in computational thinking pedagogy using agent- and object-oriented programming paradigms: entity naming strategies; token/type relations; and transitive structure changes when contrasting natural language and computer program representations.

The paper is structured in five sections. After this brief introduction, we present the essential details of the empirical study we did with a group of Brazilian students. Then we describe the method of analysis that we applied to the data. Next, we report our findings, illustrating the kinds of evidence that support them. In the last section we discuss our findings in the light of related work and indicate future directions that we want to explore.

2 A Study with Brazilian Students

We worked with a group of twenty 9th-grade students, thirteen females and seven males, between 14 and 16 years of age. They volunteered to participate in a short Scalable Game Design program [18] after class. The school is located in Niterói, 14 km away from Rio de Janeiro, across the Guanabara Bay. It is a public school whose teachers are associated with Universidade Federal Fluminense, a large public university. Most of the students in this school come from low-income communities in surrounding areas. Our group was led by a Geography teacher, also a volunteer, who had been using computers and GIS applications in his teaching.

Table 1. Participants' most liked and disliked disciplines in the curriculum

	'Which 3 disciplines you like most?'			'Which discipline you dislike most?'		
	MALES	FEMALES	ALL	MALE	FEMALES	ALL
PORTUGUESE	0%	54%	35%	29%	15%	20%
MATH	86%	31%	50%	0%	46%	30%
SCIENCE	43%	62%	55%	14%	8%	10%
PHYS ED	71%	69%	70%	0%	8%	5%
ENGLISH	14%	46%	35%	0%	8%	5%
GEOGRAPHY	86%	31%	50%	0%	0%	0%

In Table 1 we show the most liked and disliked disciplines in this group's opinion. Their preference for outdoor activities is very clear (70% said they liked Phys Ed classes). Notice the trace of gender characteristics with respect to disciplines involving languages (see preferences for Portuguese and English) and Math (see like *vs.* dislike distributions).

Table 2. Most frequent activities of participants when using computers

	3 most frequent activities when using computers		
	MALES	FEMALES	ALL
GAMES	100%	38%	60%
CHAT	57%	46%	50%
SOCIAL NETWORKS	86%	85%	85%
NEWS	14%	23%	20%
SCHOOLWORK	14%	85%	60%

In Table 2 gender characteristics stand out once more. All of the boys listed games as one of the most frequent activities when using computers, whereas just about 1/3 of the girls said so. However, almost all girls listed schoolwork as a frequent activity, against a very low percentage of boys who said so. Connecting with people in social networks was unanimously appreciated by all participants.

The project was an introduction to computational thinking with AgentSheets. It consisted of seven 2-hour weekly sessions. The goal of the program was to teach students to design a simple game related to environmental issues. For example, students built games where the player had to chase wildlife hunters, prevent forest fires, collect polluting garbage, etc. The teacher learned how to program with AgentSheets and then taught the students. Members of our research team provided technical help when needed.

The first four sessions in the project were an introduction to AgentSheets and computational thinking. Students played with AgentSheets *Frogger*, a simple version of the famous arcade game originally developed by Sega. By inspecting agent behavior in *Frogger*, the students learned the basics of Visual AgenTalk, AgentSheets programming language. They also had the opportunity to learn basic computational thinking concepts used in AgentSheets simulations, like collision, absorption, transportation, generation, diffusion, polling/counting and user input commands.

The purpose of the study was to know how these novice teenage programmers used AgentSheets expressive resources (agents' names, depictions, behavior-defining rules, etc.) to encode what they *meant* to do with their games. We aimed to achieve an in-depth understanding of the interpretive and expressive processes involved in program codification. Following a qualitative methodology, we analyzed the collected data using a combination of linguistic, semiotic and cognitive perspectives (see section 3).

At the beginning of the program, participants were asked to fill out a questionnaire with general information about themselves. All seven sessions, which took place in the school's computer lab, were videotaped. Two observers took notes and produced a short narrative of what happened in each session. In the last three sessions, we interviewed students and teacher. Most of the results reported and discussed in this paper come from these interviews and the corresponding versions of the game.

Of the twenty students who volunteered to participate, three never came to the sessions. We used data from ten out of the remaining seventeen (seven females and three males). This was mainly because some students missed the final sessions for different reasons, and because hardware problems with computers damaged critical files of two projects.

3 Method of Analysis

Our method of analysis is structured as a semiotic triangulation. The pivotal concept used in the process is that of a sign. Different semiotic theories have different definitions for it. We use Peirce's definition and sign classification system [15]. A sign is anything that some particular mind (typically a human mind) takes to stand for something else in some particular respect or context. There are three important elements in the structure of a sign: the representation (called 'representamen' in Peirce's theory), which stands for something else; the object (that which the representation stands for); and finally its assigned meaning (called 'interpretant'). Unlike other triadic theories of meaning [13], Peirce's is open-ended in the sense that the 'interpretant' is itself another sign (whose interpretation is yet another sign, and so on). This infinite digression introduces a process perspective on meaning, which can be easily traced in everyday life. It says that our interpretation of signs like words, images, scent, gestures, situations, is constantly evolving, influenced by previous meanings. The unlimited recursive process is halted and resumed for pragmatic reasons. A lack of resources like time, knowledge, motivation, interest, can cause evolutionary interpretation to stop until more resources are encountered.

Peircean Semiotics is thus especially fit to analyze how our group of students interpreted AgentSheets and gradually acquired computational concepts. Since the theory postulates meaning as evolution, what we will discuss here is a window on interpretation. The results should be taken as indications of semiotic processes that are in place, and can possibly be facilitated with new teaching strategies, new tool features, etc. They are not predictions of what will necessarily happen in similar groups and situations. The value of our research lies in revealing new aspects and perspectives in the phenomena we observed.

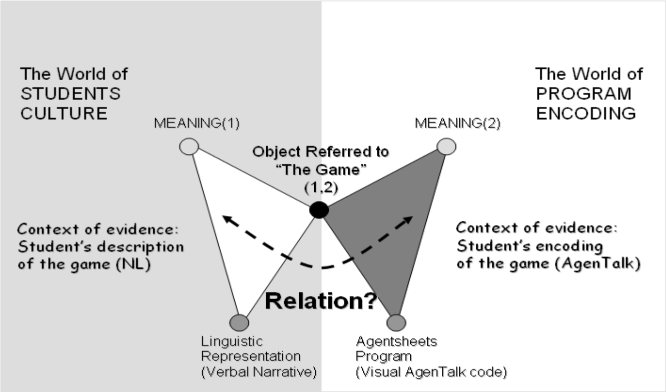

The World of STUDENTS CULTURE
The World of PROGRAM ENCODING
MEANING(1)
MEANING(2)
Object Referred to "The Game" (1,2)
Context of evidence: Student's description of the game (NL)
Context of evidence: Student's encoding of the game (AgenTalk)
Relation?
Linguistic Representation (Verbal Narrative)
Agentsheets Program (Visual AgenTalk code)

Fig. 1. A visual sketch of the triangulation process used in this research

The triangulation process is visually sketched in Fig. 1. On the left-hand side, against the shaded background, are signs that belong to the students' reality, the world of culture from which we factor out signs related to AgentSheets interface, programming language and game experience. These are represented on the right-hand side of the image. The two triangles represent sign structures, with three vertices: representation (bottom); object (center); and meaning (top). Note that triangles share their sign *object*, which is borrowed from the world of culture and used in the world of programming. The shared object is the *game* itself, a culturally-embedded computer artifact, which is mentally conceived (cognitive stance), verbally described (linguistic stance), and programmed (computational stance) in Visual AgenTalk.

To achieve this triangulation, we proceed with the following steps:

1. **A linguistic analysis of the verbal description of the game produced by students during interviews.** This analysis is based on narrative discourse, and identifies typical elements of narrative structure: characters (protagonist and antagonists); plot (temporal and causal relations among actions carried out by characters); setting (resources and obstacles); and goal (winning and end of game conditions). We paid particular attention to nouns and verbs used in verbal descriptions, and to the emerging semantic categories and relations that they indicated.
2. **A semiotic analysis of the corresponding game structure (agents' names, depictions and behavior), followed by a comparative analysis of the game execution (a player's control and perception of agent behavior during the game).** The game structure indicated abstractions and conceptual modeling constructs that the participants elaborated in the process of building the game. It also indicated their encoding choices and strategies to produce desired visual effects perceived during the game play. The comparative analysis between game structure and game execution gave us insights on how the participants realized the representational mediation between the game as an artifact and the meanings that populated their semiotic universe. For example, when the player sees "the hunter kill the monkey", the game structure may actually specify that "the monkey disappears when it sees

the hunter". This case of inverted transitivity can only be noticed and investigated if we compare program execution and structure.

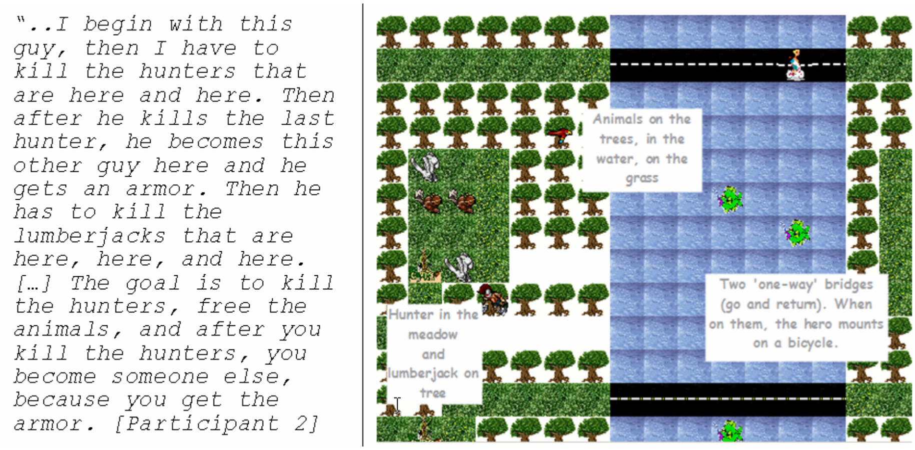
3. **A contrastive analysis of relations between meanings derived from an interpretation of verbal descriptions and an interpretation of program structure and execution**. This analysis is based on the correspondence between nouns, verbs and semantic relations in verbal narratives, on the one hand, and agents' names, depictions, behavior and effect during simulation, on the other.
4. **The final integration and interpretation of results.** In the context of computational thinking education, we focused on the semiotic resources made available and salient in AgentSheets. Our aim at this stage is to produce a cohesive characterization of how this particular group of students *makes sense* of the whole computational thinking activity supported by AgentSheets. We also want to know what this sense-making process may *mean* for researchers interested in this and related topics.

4 Findings from our Study

The ten projects showed different levels of complexity in computational thinking terms. Likewise, verbal descriptions of the game showed different levels of elaboration and verbal fluency. However, one did not go with the other. There were very cohesive and efficient verbal narratives, whose computer implementation was very rudimentary, and vice-versa. For example, Participant 2, a boy that could barely produce an understandable description of his game in Portuguese, produced a highly sophisticated program for a two-phase game, with more than 20 agents, and elaborate spatial constraints. In Figure 2, we show a translated excerpt of his verbal narrative (see left side) and a visually annotated snapshot of his game setting, showing different agents, with causal and temporal connections between them (see right side).

Notice that Participant 2 constantly switches pronouns in the verbal description (I, he, you). Notice also the frequency of use of a semantically vacuous noun ('guy') and deictic references ('here, here, and here'). Trees, meadow, grass, river, bridge – none of these visual elements present in the computer narrative were mentioned in the verbal description of the game. Neither were the bicycle and the fact that the hero of the game could only take one of the two one-way bridges to cross the river. Participant 2 had difficulties to express himself verbally all through the interviews, but he was absolutely *fluent* using another semiotic system and medium, the computer.

Many of the projects had a relatively large number of 'passive' agents (agents without behavior) that were not deployed in the game setting. This is an indication that participants who did this probably spent considerable time preparing the setting of the game, but did not get to a stage of putting all pieces together into a rich game plot. In most cases, these passive agents were not explicitly mentioned in the verbal description produced by participants. Moreover, nouns semantically associated with classes or types of entities like 'animals' and 'garbage', and with what we might refer to as a *semantic field* in itself, like 'the forest', were frequently encoded as a collection of agents. We found evidence like: raccoons, monkeys, parrots, rabbits for 'animals'; and plastic bags, cans, and other kinds of litter for 'garbage'. Different kinds of trees, meadows, flowers, trails, lakes and rivers often stood for 'forest'. A large number of agents, however, posed a harder programming challenge in the game. Defining behavior rules for each of the 'active agents', which should be at least moving around during the game, if not interacting with one another or with the player, was not only a time-consuming but also a conceptually harder task.



‟..I begin with this guy, then I have to kill the hunters that are here and here. Then after he kills the last hunter, he becomes this other guy here and he gets an armor. Then he has to kill the lumberjacks that are here, here, and here. […] The goal is to kill the hunters, free the animals, and after you kill the hunters, you become someone else, because you get the armor. [Participant 2]

Animals on the trees, in the water, on the grass
Two 'one-way' bridges (go and return). When on them, the hero mounts on a bicycle.
Hunter in the meadow and lumberjack on tree

Fig. 2. Participant 2's game versions in verbal description and computer program

During this initial phase of the project, students did not have the appropriate programming abstractions to avoid duplication of code. For example, when programming an agent's encounter with another agent, students had to define four rules (*see* up, down, left, and right), or even eight rules (to include diagonal adjacency). Consequently, in many games there was not much action going on during the game. When action was more intense on screen, this was typically the result of replicas of a particular agent, whose behavior was to 'move randomly' in specific areas of the game space. It was thus interesting but not surprising to see that the diversity and richness of meanings associated with the game setting were encoded mainly in static form, in contrast with a predominantly naïve encoding of diversity and richness in terms of behavior.

The analysis of verbal descriptions also showed that, although all games had a protagonist (the agent controlled by the player), not all of them had antagonists. For example, one of the girls described her game like this:

```
I will scatter all this garbage in the forest. [The monkey]
will have to collect it, and clean the forest. (Participant
5)
```

Notice that although there is a *goal* to be achieved (to collect all the garbage), there is no clear definition of where this garbage is coming from and how fast. Moreover, there is no definition for a winning or gaining condition. The *moral* of this story is to educate the players, showing that cleaning the forest is a *tedious* job. In a previous passage, Participant 5 said:

> In real life, it is not the monkey, but people that collect the garbage, and we should be aware that they are working [*sic*], of all work that they have to do. There are disgustingly messy people around […] They are not aware that their mess and garbage can damage Nature. (Participant 5)

Notice that this participant did not master the game *genre* as a way to communicate her message. However, the tediousness of having to clean-up other people's mess was clearly conveyed, as a *model representation* of what the participant had in mind. Therefore, we can see in her program the trace of computational thinking acquisition even though the game as such is not enjoyable.

The lack of antagonists was not necessarily a sign of tediousness. Some games had obstacles – passive agents that constrained the action of the protagonist by just *being there*. A number of participants explored the idea of a *labyrinth* in the game. The protagonist had to find his way out of it, avoiding obstacles that simply did not let the player go through. They did not destroy the player, but made him or her lose time, for example. Actually, the construction of a visual sign for a labyrinth or trail, led us to a recurring category in computer sign-making activities among this group. In AgentSheets, agents can have various *depictions* associated to them. In *Frogger*, for example, the 'frog' has two depictions, which can be associated to two different *states* of the agent: frog (normal state, apt to play); and squished frog (dead after colliding with a truck, unable to continue playing).

Different depictions can lead to useful abstractions in AgentSheets programming. For example, there are two related conditions – *stacked* and *stacked-a* – that operate at different levels of abstraction. So, if there are two depictions for the agent BRIDGE:

`'bridge'` `and 'west_side_bridge'` ,

rule 1, below, defines behavior for agent 1 when it is stacked on any agent with depiction 'west_side_bridge'. Rule 2 defines the behavior of agent 2 when stacked on any bridge, regardless of how it is depicted.

1. `If I am stacked-immediately-above an agent that looks like this` `, then I […]`
2. `If I am stacked-immediately-above a BRIDGE, then I […]`

This feature introduces a fundamental concept in computational thinking, the distinction between types and tokens, in spite of some naming constraints that may lead to equivocal interpretations. When creating a new agent, the user must give it a name and primary depiction (*e. g.* 'bridge'). Although depictions can be freely changed and added after the agent is created, its *name* cannot be changed and, by necessity, it is associated to the first depiction assigned to it (the *default* depiction). Therefore, although 'animal', for example, is an appropriate 'type sign' to refer to 'raccoon(s)', 'monkey(s)', and other *animals*, once a new agent is named 'animal', the default 'animal' depiction has to be created, leading to potential breakdowns in systematic token/type relations between agent depictions and agent names. In Figure 3, we show a comparison of object names and depictions in *Frogger*. On the left-hand side, we see a snapshot of the current version of AgentSheets Gallery of agents. On the right-hand side we show the sketch of an alternative interface design, where names and default depictions do not have to share the same identifier.

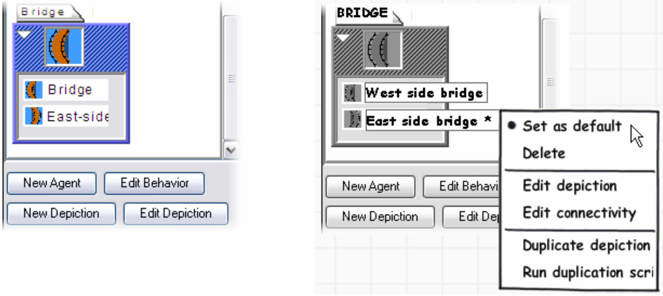


Fig. 3. Different interface alternatives to represent agent depictions in AgentSheets

In one case (left side of Figure 3), a reading of the interface leads to a breakdown with 'is-a' relationships. Although a 'west-side-bridge' *is a* 'bridge', saying that a 'bridge' *is a* 'bridge' challenges our interpretation. In the other (right side of Figure 3), the choice of an agent's name and default depiction being independent, token/type relations lend themselves to a more consistent reading (an 'east-side-bridge' *is a* 'bridge and a 'west-side-bridge' *is a* 'bridge', regardless of which one is chosen as a default depiction).

Data collected in our study clearly indicates that many students had the right intuition about type/token relations, and tried to encode it in the program. For example, the strategy used to delimit the area where the protagonist could move was often one of creating a physical boundary around it with a linguistic marker for *type*. One girl created four agents and deployed them at the edges of the game area: "top-barrier", "bottom-barrier", "left-barrier" and "right-barrier" (Participant 9). A boy surrounded the game area with four agents: "tree 1", "tree 2", "tree 3" and "tree 4". The same strategy was used by another girl (Participant 10), who created "fence 1", "fence 2", "fence 3" and "fence 4". Depictions were the same or nearly the same, showing an important meaning invariant across related agent representations.

This is an important factor to consider in computational thinking acquisition for at least two reasons. First, the presence of linguistic markers of *type signs* in students' naming strategies suggests that they were ready to deal with a higher level of abstraction in visual and textual representations of the game than the programming tools made available to them in the project allowed for. Second, a nearly opposite strategy was also found. Some students used a single agent (with single depiction) to represent the boundaries of the game area. However, interesting distinctions in behavior when the protagonist reached the edge of the space were not correctly encoded. For example, Participant 4 had difficulty to program the behavior of the player agent when encountering "leaves". She put leaves on three of the four sides of the game space. The fourth side had a different configuration – a patch of leaves separated the agent from the water. The behavior of the agent when encountering boundary leaves should be to *back up*, and when encountering the leaves next to the water should be to step on them and proceed to the water. This created a conflict in rules, and the final behavior of the agent was determined by rule order. However, the agent clearly did not behave as desired and got *stuck* when reaching certain positions on the game grid.

Another finding regarding the triangular relationship between semantic concept (signified by words appearing in verbal descriptions of the game), agent name, and agent depiction had to do with replicated agents. As mentioned above, we found cases of one-to-one and one-to-“linguistically-related many” depiction/name relations in the data. Let us call these *token-indicator* signs and *type-indicator* signs, respectively. An examination of the former shows that distinct semiotic processes were in place. In some cases, like the specification of the protagonist agent, a token-level sign had a unique instantiation in the game. This happened in all 10 games we analyzed. In other cases, token-level signs had many *replicas* instantiated in the game. For example, in most games there were agents like ‘monkey’ or ‘flower’, whose names did not so clearly represented a *class* or *type*, like those with a linguistic type marker, especially in view of its *meaning* communicated by the game setting. See Figure 4, for example, where Participant 9 represents her game setting.

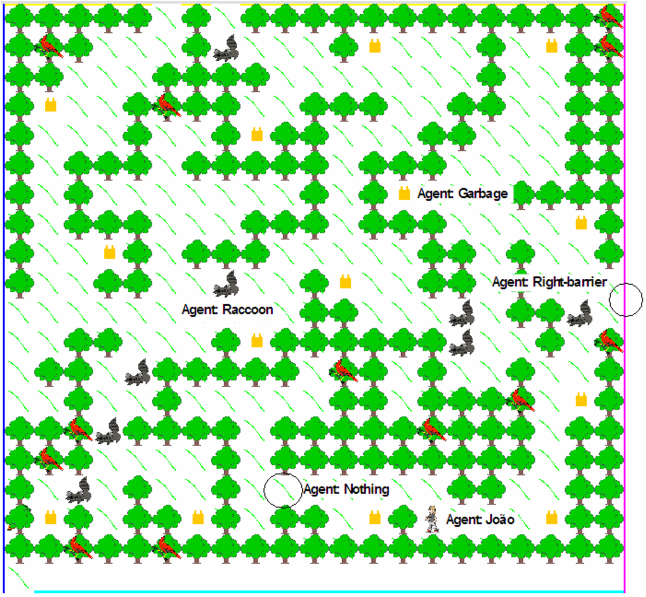


Fig. 4. Game setting visual representation with replicated agents

There are many *replicas* of raccoons and garbage on the ground, and a single instance of “João” (a proper name in Portuguese), the protagonist. Semantically, each raccoon instance is a different individual (token) of a class of animals we call ‘raccoon’. Had the game been a *model* of the animal world, the behavior of each individual raccoon should follow its own particular programming, and not be the same as is the case with *replicas*. This distinction is not important in the case of linguistically marked type signs like the already mentioned agents named as “top-barrier”, “bottom-barrier”, “left-barrier” and “right-barrier”. Although the four *barriers* are composed of aligned *replicas* of the corresponding barrier agent (the right barrier is actually an array of 20 instances of “right-barrier” agent depictions), our interpretation of the visual sign integrates all instances of barrier into a single conceptual unit, identified by its semantic role (a boundary). So, we actually *expect* that all behaviors will be the same, which reinforces the interpreted meaning assigned to the unified object. There is also a wonderful semiotic choice in the game code that generates the setting shown in Figure 4. The *name* of the agent depicted as a diagonal line, replicated to form the *path* in the forest where the player can navigate and collect plastic bags, is “nothing” (‘nada’ in Portuguese). This is an indication of how Participant 9 conceptualized the game – the protagonist can walk in places where there is *nothing* to impede it. However, ‘nothing’ must be visible, must have a depiction, whose appearance and name really do not matter. This shows that this participant had an intuition of formal representation principles involved in computational thinking (*i. e.* that computer objects must be linguistically signified).

Another semiotic process that Participant 9 (among others) gives us a chance to illustrate is that the computer codification and the verbal account of the same referent object (see Figure 1) prompt for signs of very different nature. Her verbal description mentioned a labyrinth (“The goal of the game [is] to go through this labyrinth.”), which is not directly encoded as an agent – active or passive – but is clearly shown when the program is executed. There are various dead ends in the path that “João” can tread as the game evolves. So the labyrinth is signified by depictions and behavior of many other agents in the game, but it is never *named*. This is a complex programming skill that must be acquired and mastered if we want students to be able to build more sophisticated computational models of various kinds of phenomena – real or imaginary.

The last finding we present in this section is related to semantic transitivity. When analyzing verbal descriptions of the game, we saw that, in general, students produced evidence of *naturalistic* transitivity. That is, when talking about causal relations involved in agent interactions, most of the time they used preferential transitive structures in Brazilian Portuguese to describe what happened (*e. g.* “he collects the garbage” or “he kills the hunter”). In some cases, they also used intransitive structures that left causal relations partially undefined (*e. g.* “he meets the ranger and dies” instead of “he meets the ranger and is killed” or, alternative, “he meets the ranger, and the ranger kills him”). Pragmatic implicature in the text made it easy to recover the causal chain in this case. The most probable reason why the agent died when meeting the ranger was that *the ranger did something to it* that caused is death, and not that it committed suicide in the sequence of the meeting, for example. Nevertheless, in all but one case, students built games with at least one occurrence of causal relations *programmed* in reverse order.

Inverted transitivity is present in *Frogger*, the game that was used in the first four sessions of the project, and illustrates a computational thinking pattern called diffusion [1]. For example, the following behavior rules are defined for the FROG:

```
1  If STACKED (immediately above , 🏝) Then SAY (Cannot walk
   over a turtle maker. That's cheating!), and ERASE ( ■ ),
   and STOP-SIMULATION ()
```

2 If STACKED (immediately above , 🟦) Then `SAY (I cannot swim!), and ERASE ( ▪ ), and STOP-SIMULATION ()`

Line 1 above is saying that if the frog is on top of (the depiction of) the turtle maker, then: the frog says, “Cannot walk over a turtle maker. That’s cheating!”; erases the object in the same location where it stands (*i. e.* the frog erases itself); and stops the simulation. Line 2 says that if the frog is on top of (the depiction of) the river (a patch of blue color), then: the frog says “I cannot swim!”; erases the object in the same location where it stands (*i. e.* the frog erases itself); and stops the simulation.

The presence of these signs in the program encoding gives us the opportunity to appreciate aspects of *code patterns and reuse*, a prime feature in object-oriented programming. The frog erasing itself and stopping the simulations lends itself to the interpretation that this is a *suicidal move*, and makes perfect sense in the context of the game. The frog is the semantic agent of all actions in line 2: *it* says that it cannot swim, *it* disappears, and *it* causes the game to stop. Now, when looking at line 1 it seems semantically inconsistent to suppose that the frog will commit suicide by stepping on an island (the surprising depiction of a “turtle maker”), or that it will reprimand itself for cheating. Maybe the island should magically eliminate the unknowing frog and stop the game after shouting ‘You cannot walk over a turtle maker. That’s cheating!’. However, although this would probably be more consistent with the fantastic logic of the game, it would have the programming cost of encoding a new kind of rule for *turtle makers*: if stacked immediately *below* the frog, then say [something], erase stacked object at location (frog) and stop the simulation. Another cost would be the effect of diffusion in this case: to transfer the encoding of frog behavior to other agents, preventing the programmer from seeing all threats and opportunities associated with the frog in the same chunk of code.

Students in our group showed us numerous cases of inverted semantic transitivity of actions. We give two illustrations. Participant 1 encoded behavior stating that when the player saw the ranger, the player erased itself and stopped simulation. Participant 8, who followed the Space Invaders model in his game, had a canon shooting water in the sky to destroy flying lanterns that fell on trees and caused big fires. His encoding, however, transferred transitivity from the shooting to the lanterns, and defined that if the lantern met water, then it erased itself. Although in this case one could argue that a switch in perspective might, indeed, allow us to describe the action from the lantern point of view, there is no way to select one perspective and keep with it throughout the whole process. Switching from one to the other is perfectly possible and, as noted before, potentially advantageous for program maintenance. We must ask ourselves if there are cognitive issues associated with this when it comes to building consistent computational models. Which representation must be taken as a *model* of a given phenomenon: sign structures in program encoding or the visual effects they produce?

5 Discussion and Future Work

Our work is related in different ways to previous research. It is connected to educational projects for teaching computational thinking not only with AgentSheets [18], but also with other technologies such as Alice [8, 3] and Scratch [9, 19], for example.

The innovative aspects of our research come from the semiotic approach adopted in this study, and from the context of our research. As far as we know, this is the first computational thinking project using AgentSheets in a Brazilian school.

Our work inherits from all previous work with AgentSheets a particular emphasis on the acquisition of computational thinking, rather than programming skills. We do not focus on how students used programming elements such as variables, loops, conditionals, and the like. We analyze signs, representations and meanings, following the semiotic line from the students' psychological and cultural world into the world of computing. This relates to previous research on cognitive dimensions of notational systems. For example, our analysis of token/type relations corresponds closely to Blackwell and Green's [2] abstraction and resistance to change (viscosity) dimensions. As mentioned before, there is a cost in programming behavior rules for similar but not identical agents (*e.g.* fence and barrier agents surrounding the game space). If the students had more readily available abstraction mechanisms (which are provided by AgentSheets, but require more sophisticated computational thinking skills), maybe they would have encoded repetitive agent behavior differently. Cases of premature commitment were also found. Because agents' names cannot be changed in AgentSheets, there were inconsistent relations between agents' names and depictions. Two additional cognitive dimensions were observed in the data: visibility and hidden dependencies. For example, in our discussion of inverted transitivity we remarked that although the visual effects of the underlying program might suggest that Agent A's action caused some effect on Agent B, the encoded rules might actually state that Agent B was the one acting in the presence of Agent A. In the textual specification of the program, the relation between A and B is thus invisible when we look at Agent A's behavior. There is a hidden dependency with Agent B.

Agents' behavior can be represented in encapsulated form (following the traditional object-oriented programming paradigm) or diffused form (crosscutting various agents). As Repenning has previously shown, there are computational advantages in programming AgentSheets simulations in the opposite way as prescribed by OOP [17]. He calls it 'programming with antiobjects', and discusses efficiency gains when programming collaborative behavior of agents on a platform, for example. Interactions among them can be encoded as *platform behavior* rules, leading to an impressive reduction of program complexity. The point to make in our analysis is that we should have a clear notion of which aspect(s) of computational thinking we want to explore in the teaching – the simulation behavior or the underlying rules and structures that define it? Which one stands for a model of the phenomenon it refers to in culture, nature, or an individual's imagination? Which one is a *sign* of the phenomenon?

Semiotically speaking, both are, and one stands for the other. This perspective somehow *dissolves* the tension discussed by Frasca [6], that narratives are not an appropriate model for games. He claims that simulations are a better model because they are *active* representations, and not *passive* text. Juul [7] raises a similar point, although he does not contend the narrative model for games. He remarks that whereas narratives take a time perspective that is typically oriented towards the past (literally speaking, one cannot *narrate* the future), games are oriented towards the future. They unfold into the future as the players interact with it. There are interesting ontological switches in this perspective if we conjecture that the reason why we do not *narrate* the future is that we cannot *predict* it. Actually, when the future *is* *predictable*, narratives like “I will open my hand, drop the mirror and it will break into pieces on the floor” are completely acceptable. Since computer games are formally programmed, they become considerably predictable. Thus, a narrative into the future is not as constraining as Frasca believes. We may have a problem, however, if the narrative is rooted in the formal specification of the game, that is, in the linguistic stance (rules of behavior and agent configurations). In this case, inverted transitivity may again turn into an issue. The essence of the argument has been captured by Mor and Noss [10], who speak of programming as mathematical narrative. Their research explores how narratives can be used to construct mathematical knowledge and encode it formally, in an environment that resists ambiguity. Can narratives be used as scaffolds to acquire formal computing knowledge?

In a way, Myer’s work on natural programming [11, 14] is a positive answer to the question. Natural language is not formal and precise, but it is the richest means of expression that we have. The various kinds of mechanisms that can be used to control focus, switch perspectives, refer to real and imaginary things, elicit verbal behavior from others, trigger semantic associations – all of these and more support and develop human mental activity. Thus, it is not surprising to see that programming languages actually incorporate – even if partially – a number of these mechanisms in their designs. Should or could AgentSheets increase its semiotic power by, for example, supporting aspect-oriented programming [5]? This might create interesting new possibilities to deal with inverted transitivity in novice and advanced programming projects. A separation of concerns and perspectives might for example help us keep with direct transitivity in a domain model (logic) perspective, while allowing for diffusion in a program architecture (software engineering) perspective. The two perspectives depend on related but not identical computational thinking skills.

In a thoughtful exploration of aspectual control, Sedig and co-authors [20] report interesting results with three alternative styles of direct manipulation interfaces for a Tangram game to teach Euclidean geometry. With the first, smaller Tangram polygons can be freely dragged, dropped and rotated to fill a larger polygonal area. The acquired skills at this stage have to do with spatial reasoning, but the learners develop a model that is not consistent with Euclidean geometry. For instance, they believe that translation and rotation are just physically constrained move and turn operations. With the second interface style, learners cannot manipulate polygons directly; they must operate on specific visual controls representing axes, angles, vertices, and so on. Ghost images show a preview of the effects achieved by control manipulations. As a result, at this stage, learners begin to associate geometric parameters of translation and rotation with certain visual effects. The third style of direct manipulation eliminates the ghost image and requires that the learner be able to preview mentally (*i. e.* to calculate) the effects of geometric variable and operation control manipulations. This involves mental representations that resemble an algebraic formulation of geometric functions. It is only at this last stage, according to Sedig and co-authors, that learners really understand Euclidean geometry concepts that can be used resolve Tangram challenges.

De Souza and Sedig [4] have shown that there are powerful semiotic mechanisms in place at each stage of abstraction in the geometric game. Different styles of direct manipulation actually correspond to different types of signification systems. Increasing levels of formality associated with the meaning of visual objects actually correspond to a progression in semiotic categories that evolve from the perceptual (first style), to the associative (second style), and then to the formal (third style) aspects of a geometric *model* representation.

Bringing the above to the context of acquiring computational thinking skills with AgentSheets, students can begin to build games focusing only on the *perceptual qualities* observed when the game is executed. At this stage, which internal representations and structures are used does not matter, as longs as the students experience the power of producing and interacting with the game. At the next stage in the learning, students can begin to explore how different internal representations can produce the same perceptual qualities when the game is executed. The *association* of alternative internal representations with the same external effects can be used to introduce important aspects of computation, like efficiency for example. Finally, at an advanced stage, students should be able to focus on internal representations *per se*, exploring patterns of structure and relations that can help them learn *qualitative aspects of programming* (like factoring, reuse, etc.). The main advantage of working with different *signs* at each stage is to make the students realize that there are, indeed, different *narratives* that can be produced to describe *the game*. This would be a powerful and yet natural encounter with the fact that a single program *signifies* different things when looked from the outside, from the inside, or from a crosscut perspective. A vivid understanding of each perspective would certainly be a major step in computational thinking education, which semiotic theories can contribute to achieve by informing the design of alternative AgentSheets interfaces styles that control the use of signs at different learning stages.

Our next research steps go in this direction. We want to take advantage of the semiotic richness of AgentSheets and explore alternative teaching strategies, in order to see how computer signification systems can be better learned and used by middle school students. The fact that we are using a semiotic perspective will also allow us to track potential cultural differences between Brazilian and non-Brazilian students participating in Scalable Game Design projects in various countries. At a later stage, we can select the best signification systems we found and use them to design alternative interface styles to support scaffolded teaching strategies with AgentSheets.

Acknowledgments. We thank the director, teachers and students of COLUNI for much appreciated collaboration in this project. This material is based in part upon work supported by the National Science Foundation under Grant Numbers 0833612 and DMI-0712571. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. Clarisse de Souza is partially supported by grants from FAPERJ (E-26/102.400/2009) and CNPq (308964/2006-3). Cristina Garcia is also partially supported by CNPq.

References

1. Basawapatna, A.R., Koh, K.H., Repenning, A.: Using scalable game design to teach computer science from middle school to graduate school. In: Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2010), pp. 224–228. ACM, New York (2010)

2. Blackwell, A.F., Green, T.R.G.: Notational systems - the Cognitive Dimensions of Notations framework. In: Carroll, J.M. (ed.) HCI Models, Theories and Frameworks: Toward a Multidisciplinary Science, pp. 103–134. Morgan Kaufmann, San Francisco (2003)
3. Cooper, S., Dann, W., Pausch, R.: Teaching objects-first in introductory computer science. In: Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2003), pp. 191–195. ACM, New York (2003)
4. de Souza, C.S., Sedig, K.: Semiotic considerations on direct concept manipulation as a distinct interface style for learnware. In: IHC 2001 - IV Workshop de Fatores Humanos em Sistemas Computacionais, pp. 229–241. SBC, Porto Alegre (2001)
5. Elrad, T., Filman, R.E., Bader, A.: Aspect-Oriented Programming. Communications of the ACM 44(10), 28–32 (2001)
6. Frasca, G.: Simulation versus narrative: introduction to ludology. In: Wolf, M.J.P., Perron, B. (eds.) The Video Game Theory Reader, pp. 221–235. Routledge, London (2003)
7. Juul, J.: Games telling Stories? A brief note on games and narratives. Game Studies 1(1) (2001), Online at http://www.gamestudies.org/0101/juul-gts/
8. Kelleher, C., Pausch, R.: Using storytelling to motivate programming. Communications of the ACM 50(7), 58–64 (2007)
9. Maloney, J.H., Peppler, K., Kafai, Y., Resnick, M., Rusk, N.: Programming by choice: urban youth learning programming with scratch. SIGCSE Bulletin 40(1), 367–371 (2008)
10. Mor, Y., Noss, R.: Programming as mathematical narrative. Int. J. Continuing Engineering Education and Life-Long Learning 18(2), 214–233 (2008)
11. Myers, B.A., Pane, J.F., Ko, A.: Natural programming languages and environments. Communications of the ACM 47(9), 47–52 (2004)
12. National Research Council Committee for the Workshops on Computational Thinking (2010) Report of a Workshop on The Scope and Nature of Computational Thinking, Online at http://www.nap.edu/catalog/12840.html
13. Ogden, C.K., Richards, I.A.: The meaning of meaning, 8th edn. Harcourt, Brace & World, Inc., New York (1989)
14. Pane, J.F., Ratanamahatana, C.A., Myers, B.A.: Studying the language and structure in non-programmers' solutions to programming problems. Int. J. Human-Computer Studies 54(2), 237–264 (2001)
15. Peirce, C.S.: The Essential Peirce, Selected Philosophical Writings, vol. 1, 2, Edited by Houser, N., Kloesel, C.J.W. Indiana University Press, Bloomington (1992, 1998)
16. Repenning, A., Ioannidou, A.: Agent-Based End-User Development. Communications of the ACM 47(9), 43–46 (2004)
17. Repenning, A.: Collaborative diffusion: programming antiobjects. In: Companion to the 21st ACM SIGPLAN Symposium on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA 2006), pp. 574–585. ACM, New York (2006)
18. Repenning, A., Webb, D., Ioannidou, A.: Scalable game design and the development of a checklist for getting computational thinking into public schools. In: Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE 2010), pp. 265–269. ACM, New York (2010)
19. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y.: Scratch: programming for all. Commununications of the ACM 52(11), 60–67 (2009)
20. Sedig, K., Klawe, M., Westrom, M.: The Role of Interface Manipulation Style and Scaffolding on Cognition and Concept Learning in Learnware. ACM Transactions on Computer-Human Interaction 8(1), 34–59 (2001)

Where Are My Intelligent Assistant's Mistakes? A Systematic Testing Approach

Todd Kulesza[1], Margaret Burnett[1], Simone Stumpf [2],
Weng-Keen Wong[1], Shubhomoy Das[1], Alex Groce[1],
Amber Shinsel[1], Forrest Bice[1], and Kevin McIntosh[1]

[1] School of EECS, Kelley Engr. Center, Oregon State University, Corvallis, OR 97331
{kuleszto,burnett,wong,dassh,
alex,shinsela,bice,mcintoke}@eecs.oregonstate.edu
[2] Centre for HCI Design, City University London, Northampton Square, London EC1V 0HB
Simone.Stumpf.1@city.ac.uk

Abstract. Intelligent assistants are handling increasingly critical tasks, but until now, end users have had no way to systematically assess where their assistants make mistakes. For some intelligent assistants, this is a serious problem: if the assistant is doing work that is important, such as assisting with qualitative research or monitoring an elderly parent's safety, the user may pay a high cost for unnoticed mistakes. This paper addresses the problem with WYSIWYT/ML (What You See Is What You Test for Machine Learning), a human/computer partnership that enables end users to systematically test intelligent assistants. Our empirical evaluation shows that WYSIWYT/ML helped end users find assistants' mistakes significantly more effectively than ad hoc testing. Not only did it allow users to assess an assistant's work on an average of 117 predictions in only 10 minutes, it also scaled to a much larger data set, assessing an assistant's work on 623 out of 1,448 predictions using only the users' original 10 minutes' testing effort.

Keywords: Intelligent assistants, end-user programming, end-user development, end-user software engineering, testing, machine learning.

1 Introduction

When using a customized intelligent assistant, how can an end user assess whether and in what circumstances to rely on its work?

Although this may seem at first glance to be merely a matter of providing live feedback, assessment cannot be treated so superficially when the assistant is performing a critical task. Yet until now, there has been no way for end users to *systematically* assess whether and how their customized intelligent assistants need to be mistrusted or fixed. Instead, the mechanisms available for user assessment have been strictly ad hoc: users have had only their gut reactions to what they serendipitously happen to notice.

In their perspectives on the future of end-user development, Klann et al. pointed to the need both for intelligent customizations and quality control in end-user development

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 171–186, 2011.

[15]. In addition to Klann et al.'s arguments, there are at least three reasons why end-user assessment of today's customized assistants has become of key importance. First, no machine learning technique can yet prevent an intelligent assistant from making any mistakes. Since machine learning algorithms try to learn a concept from a finite sample of training data, issues like overfitting and the algorithm's inductive bias prevent an assistant from being 100% correct over future data. For instance, in [29], a good assistant is only 80-90% accurate.

Second, today's intelligent assistants are taking on increasingly important roles—roles in which, if the assistant goes awry, the user may bear significant costs and/or risks. For example, Gmail's new priority inbox decides which e-mail messages busy people can and cannot delay reading [10]. Other kinds of emerging assistants are moving toward helping with research itself, such as qualitatively "coding" (categorizing) natural language text [18]. Assistants are even approaching intelligent "aging-in-place" monitoring of safety status to enable geographically distant caregivers to support their aging parents [26] without being personally nearby. In this paper, we focus on end users who are willing to spend a modest amount of effort to assess assistants doing these kinds of critical tasks.

Third, if an assistant is making mistakes that are critical, the user may want to fix ("debug") the assistant, but effective debugging heavily relies on effective testing—the user needs to determine *where* the assistant's mistakes are, *when* their debugging efforts have fixed the mistakes, and *when* their previous testing and/or debugging may need to be revisited. Therefore, as we will explain in the next section, this paper maps the question of end-user assessment of a customized intelligent assistant to an *end-user testing* problem.

We thus present a human/computer partnership, inspired by the What You See Is What You Test (WYSIWYT) end-user testing methodology for spreadsheets [5, 25]. In our approach (WYSIWYT/ML), the system (1) *advises* the user about which predictions to test, then (2) *contributes* more tests "like" the user's, (3) *measures* how much of the assistant's reasoning has been tested, and (4) continually *monitors* over time whether previous testing still "covers" new behaviors the assistant has learned.

This paper makes the following contributions:

- We show how end-user assessment of intelligent assistants can be mapped to testing concepts. This mapping opens the door to potentially applying prior research on software testing to assistant assessment.
- We present our WYSIWYT/ML approach for helping users find *where* their assistant's mistakes are and monitoring *when* a previously reliable assistant may have gone astray.
- We present the first empirical evaluation of an end-user testing approach for assessing a user's evolving intelligent assistant.

Our empirical results showed significant evidence of the superiority of systematic testing in terms of efficiency and effectiveness—by one measure, improving users' efficiency by a factor of 10. These results strongly support the viability of this new method for end users to assess *whether* and *when* to rely on their intelligent assistants.

2 Intelligent Assistant Assessment as a Testing Problem

In this section, we show how assessing whether and when an intelligent assistant's outputs are right or wrong can be mapped to software testing. We adopt our terminology from the general literature of software testing [3], and in particular from the formulation of previous end-user testing problems [25].

According to the latest IEEE Standard [14], *testing* is "the process of [running] a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system or component"—i.e., running the program in a particular way (e.g., with given inputs) and evaluating the outputs.

The assistant is obviously a program, but it is an unusual kind of program in that it was, in part, automatically generated. Specifically, the intelligent assistants of interest to us are text classification assistants that output a single label for each textual input—in this domain, the programming process is as follows. First, the machine learning expert writes the assistant shell and learning algorithm, and tests or otherwise validates them to his or her satisfaction. The expert then runs the algorithm with an initial set of *training data* (here, labeled text examples) to automatically generate the first version of the assistant, which is the first version of the *program*. The assistant is then deployed to the end user's desktop.

At this point, the assistant's primary job, like that of most programs, is to read inputs (here, unlabeled text) and produce output (here, a label for that text). But unlike other programs, the assistant has a second job: to gather new training data from its user's actions to learn new and better logic—the equivalent of automatically generating a *new program*. Note that the machine learning expert is no longer present to test this new program—the newest version of the assistant learned behaviors from its specific end user *after* it had been deployed to the user's desktop. Thus, the end user (acting as an oracle [3]) is the only one present to test the program.

Given such a program, many of the testing concepts defined in Rothermel et al. [25] can be straightforwardly applied to our domain. A *test case* is the combination of an input (unlabeled text) and its output (a label). Given a test case that the program has executed, a *test* is an explicit decision by the end user about whether the output is correct for that test case's input. If the user decides that the output is not correct, this is (at least in the end user's eyes) evidence of a *bug* in the assistant's reasoning.

Testing would not be viable if every possible input/output pair must be tested individually—the space of all possible inputs is usually intractably large or possibly infinite. One solution has been to use the notion of coverage [3] to measure whether "enough" testing has been done. Along this line, consider a partitioning scheme that divides inputs into "similar" groups by some measure of similarity. A test case can then be said to *cover* all current (and future) input/output pairs for which the inputs are in the same group as the test case's input, and the outputs equal the test case's output.

Given these definitions, *systematic testing* differs in two important ways from the ad hoc testing that comes by serendipitously observing correct/incorrect behaviors: systematic testing has a measure (coverage) for ascertaining how "tested" the program is, and it provides a way to identify which test cases can increase that measure. In the spreadsheet paradigm, systematic testing by end users has been shown to be significantly more effective than ad hoc testing [5].

Finally, it is worth discussing how testing and debugging, while related, are distinctly separate activities. Testing, as we have explained, evaluates whether a program's outputs are right or wrong, whereas *debugging* is the act of actually fixing the program. Even without precisely mapping debugging of assistants to classic debugging (which is beyond the scope of this paper), it is clear that testing contributes to two phases that have been identified for debugging [19]: it contributes to the debugging phase of *finding* the bug by showing an instance of how/where a program is failing, and also contributes to the debugging phase of *validation* of whether the program has now stopped failing in that way. We envision our testing approach as contributing to both of these aspects of debugging.

3 Related Work

Testing of intelligent assistants is often done pre-deployment by machine learning specialists via statistical methods [13]. Such methods do not substitute for *end users*' assessment of their assistants because pre-deployment evaluation cannot assess suitability of after-deployment customizations to a particular user.

Some statistical debugging, however, can be automatically carried out after deployment. Research in machine learning has led to *active learning*, whereby an assistant can request the user to label the most informative training examples during the learning process [28]. Although one of our WYSIWYT/ML methods (Confidence) is sometimes used in active learning, most of our methods differ from active learning's. Our approach complements debugging techniques such as active learning, allowing the user (not the intelligent assistant) to assess whether and when the assistant is reliable.

Statistical outlier finding has been used in end-user programming settings for assessment, such as detecting errors in text editing macros [22], inferring formats from a set of unlabeled examples [27], and to monitor on-line data feeds in web-based applications for erroneous inputs [24]. These approaches use statistical analysis and interactive techniques to direct end-user programmers' attention to potentially problematic values, helping them find places in their programs to fix. Our approach also uses outlier finding, but does so as just one part of a larger approach that also systematically measures how much more assessment needs to be done.

Systematic testing for end users was pioneered by the *What You See Is What You Test* approach (WYSIWYT) for spreadsheet users [25]. To alleviate the need for users to conjure values for testing spreadsheet data, "Help Me Test" capabilities were added; these either dynamically generate suitable test values [7] or back-propagate constraints on cell values [1]. WYSIWYT inspired our approach in concept, but our under-the-hood reasoning about test prioritization and coverage are based on statistical properties of the assistant's behavior, rather than WYSIWYT's "white box" use of source code structure. Also, rather than helping users conjure new values to test, our approach instead aims to help users focus on just the right fraction of existing data to find important errors quickly.

To support end users' interactions with intelligent assistants, recent work has explored methods for explaining the reasons underlying an assistant's predictions. Such explanations have taken forms as diverse as *why...* and *why not...* descriptions of the assistant's logic [17, 20], visual depictions of the assistant's known correct predictions versus its known failures [29], and electronic "door tags" displaying predictions of worker interruptibility with the reasons (e.g., "talking detected") [31]. As a basis for creating explanations, researchers have also investigated the types of information users want before assessing the trustworthiness of an intelligent agent [9, 18]. Recent work by Lim and Dey has resulted in a toolkit for applications to generate explanations for popular machine learning systems [21], and a few systems add debugging capabilities to explanations [17, 18]. Our approach for supporting systematic assessment of intelligent assistants is intended as a complement to explanation and debugging approaches like these.

4 The WYSIWYT/ML Approach

We have explained that without systematic testing, a user is left with only the ability to assess ad-hoc the assistant's predictions that they happen to notice. Ad-hoc testing does not help the user pick *which* items to test, nor does it help the user decide *how much more* testing should be done. WYSIWYT/ML targets both issues for situations in which an assistant's mistakes carry high risks or high costs for the user.

One such high-risk/high-cost situation is qualitative "coding" of verbal transcript data (a common HCI research task), in which empirical analysts segment written transcripts and categorize each segment. This is a labor-intensive activity requiring days to weeks of time—but what if an assistant could do part of this work (e.g., [18])? For example, suppose ethnographer Adam has an intelligent assistant that learns to code the way Adam does; the assistant could then finish coding Adam's transcripts. But Adam's research results may be invalid if the assistant's work is wrong, so he needs to assess where the assistant makes significant mistakes.

We prototyped WYSIWYT/ML as part of an intelligent "coding" assistant that classifies text messages, similar to Adam's hypothetical coding assistant. The assistant in our prototypes makes its predictions using a support vector machine, but the algorithm is not important—WYSIWYT/ML works with *any* algorithm (or accompanying feature set) that produces the information needed by the test prioritization methods described shortly.

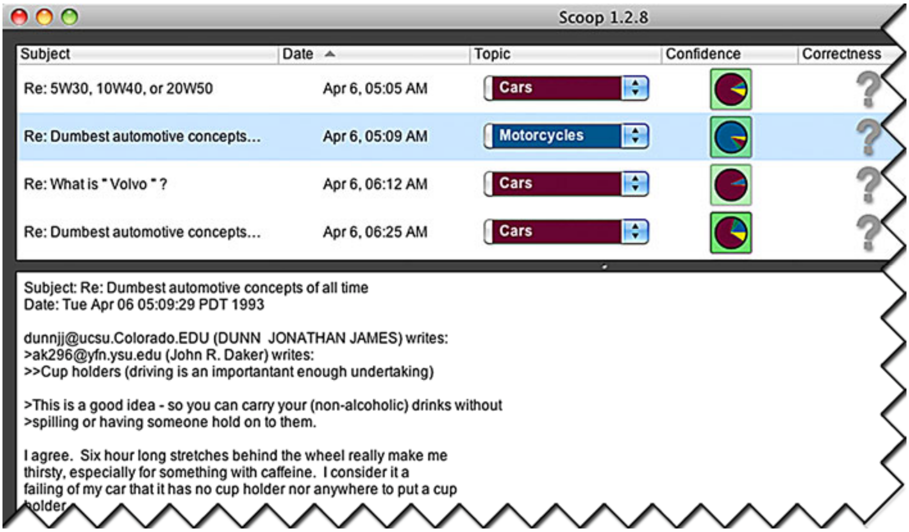
4.1 How WYSIWYT/ML and Adam Work Together

Two Use-Cases. Given an intelligent classification assistant, WYSIWYT/ML's mission is to help the user assess its accuracy during two use cases.

Use case *UC-1*: In the assistant's early days, can Adam rely on it? After his assistant has been initially trained, Adam can use WYSIWYT/ML to decide whether it classifies messages consistently enough for his purposes. To minimize time spent finding the assistant's mistakes, WYSIWYT/ML advises him *which* messages the assistant believes it is weakest at classifying.

Use case *UC-2*: As the assistant continues to customize itself, can Adam *still* rely on it? As the assistant continues to learn and/or new messages arrive, WYSIWYT/ML keeps track of whether the assistant is working on messages very similar to (and sharing the same output label as) those previously tested, or whether the assistant is now making predictions unlike those tested earlier. If the assistant is behaving differently than before, test coverage will be much lower and Adam might decide to systematically test some of the assistant's new work. WYSIWYT/ML helps him target these new predictions.

To support these use-cases, WYSIWYT/ML performs four functions: (1) it *advises* (prioritizes) which predictions to test, (2) it *contributes* tests, (3) it *measures* coverage, and (4) it *monitors* for coverage changes.



Subject	Date ▲	Topic	Confidence	Correctness
Re: 5W30, 10W40, or 20W50	Apr 6, 05:05 AM	Cars		?
Re: Dumbest automotive concepts...	Apr 6, 05:09 AM	Motorcycles		?
Re: What is "Volvo"?	Apr 6, 06:12 AM	Cars		?
Re: Dumbest automotive concepts...	Apr 6, 06:25 AM	Cars		?

Subject: Re: Dumbest automotive concepts of all time
Date: Tue Apr 06 05:09:29 PDT 1993

dunnjj@ucsu.Colorado.EDU (DUNN JONATHAN JAMES) writes:
>ak296@yfn.ysu.edu (John R. Daker) writes:
>>Cup holders (driving is an importantant enough undertaking)

>This is a good idea - so you can carry your (non-alcoholic) drinks without
>spilling or having someone hold on to them.

I agree. Six hour long stretches behind the wheel really make me
thirsty, especially for something with caffeine. I consider it a
failing of my car that it has no cup holder nor anywhere to put a cup
holder

Fig. 1. The WYSIWYT/ML prototype. This variant uses the Confidence method.

WYSIWYT/ML Prioritizes Tests. WYSIWYT/ML prioritizes the assistant's topic predictions that are most likely to be wrong, and communicates these prioritizations using saturated green squares to draw Adam's eye (e.g., Figure 1, fourth message). The prioritizations may not be perfect, but they are only intended to be *advisory*; Adam is free to test any messages he wants, not just ones the system suggests.

To select prioritization methods, we first ran offline experiments using a "gold standard" oracle (rather than real users) to allow for numerous experiment runs. These experiments compared five candidate prioritization methods against randomization (where *Random* represents the statistical likelihood of finding mistakes). We selected the three best-performing methods, all of which outperformed Random: *Confidence*, *Similarity*, and *Relevance*.

The *Confidence* method leverages the assistant's knowledge of its own weaknesses, prioritizing messages based on the assistant's certainty that the topic it predicted is correct. (This is also a method used by active learning [28].) The higher the uncertainty, the more saturated the green square (Figure 1, Confidence column). Within the square, WYSIWYT/ML "explains" Confidence prioritizations using a pie chart (Figure 2, left). Each pie slice represents the probability that the message belongs to that slice's topic: a pie with evenly sized slices means the assistant thinks each topic is equally probable (thus, testing it is a high priority).

The *Similarity* method selects "oddball" messages—those least similar to the data the assistant has learned from. The rationale is that if the assistant has never before seen anything like this message, it is less likely to know how to predict its topic. We measure this via cosine similarity [2], which is frequently used in information retrieval systems; here, it measures co-occurrences of the same words in different messages. A "fishbowl" explains this method's priority, with the amount of "water" in the fishbowl representing how unique the message is compared to messages on which the assistant trained (Figure 2, middle). A full fishbowl means the message is very unique (compared to the assistant's training set), and thus high priority.

The *Relevance* method is based on the premise that messages without useful words may not contain enough information for the assistant to accurately predict a topic. In machine learning parlance, useful words have high *information gain* (i.e., the words that contribute the most to the assistant's ability to predict the topic). We used the top 20 words from the messages the assistant learned from, then prioritized messages by the *lack* of these relevant words. Our prototype uses the number of relevant words (0 to 20) to explain the reason for the message's priority (Figure 2, right), with the lowest numbers receiving the highest priorities.

In our offline tests (without users), the Confidence method was the most effective: its high-priority tests were very successful at identifying flaws in an assistant's predictions, even when the assistant was 80% accurate. The Similarity and Relevance methods did not highlight as many bugs, but they outperformed Confidence in revealing hard-to-find bugs: items the assistant thought it was right about (predicted confidently), but which were wrong. We thus implemented all three, so as to empirically determine which is the most effective with real users.

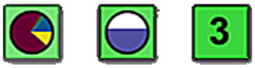


Fig. 2. The Confidence (left), Similarity (middle), and Relevance (right) visualizations.

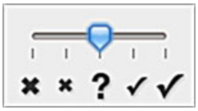


Fig. 3. A user can mark a predicted topic *wrong*, *maybe wrong*, *maybe right*, or *right* (or "?" to revert to untested). Prior research found these four choices to be very useful in spreadsheet testing [12].

Use-Case UC-1: Adam Tests His Assistant. When Adam wants to assess his assistant, he can pick a message and judge (i.e., test) the assistant's prediction. He can pick any message: one of WYSIWYT/ML's suggestions, or some different message if he prefers. Adam then communicates his judgment by clicking a *check* if it is correct or an *X* if it is incorrect, as in Figure 3. If a topic prediction is wrong, Adam has the option of selecting the correct topic—our prototype treats this as a shortcut for marking the existing topic as "wrong", making the topic change, and then marking the new topic as "right".

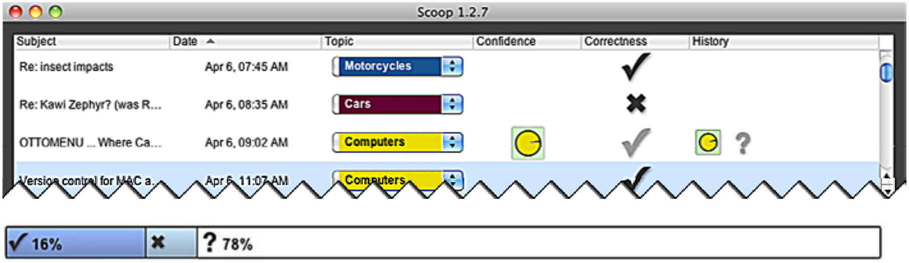
WYSIWYT/ML then *contributes* to Adam's testing effort: when Adam tests a message, WYSWYT/ML automatically infers the same judgment upon similar messages. These automated judgments constitute *inferred tests*.

To contribute these inferred tests, our approach computes the cosine similarity of the message Adam just tested with each untested message sharing the same predicted topic. WYSWYT/ML then marks very similar messages (i.e., scoring above a cosine similarity threshold of 0.05) as *approved* or *disapproved* by the assistant. The automatically inferred assessments are shown with gray check marks and X marks in the Correctness column (Figure 4, top), allowing Adam to differentiate his own explicit judgments from those automatically inferred by WYSIWYT/ML. Of course, Adam is free to review (and if necessary, fix) any inferred assessments—in fact, most of our study's participants started out doing exactly this.

WYSIWYT/ML's third functionality is *measuring* test coverage: how many of the assistant's predictions have been tested by Adam and the inferred tests together. A test coverage bar (Figure 4, bottom) keeps Adam informed of this measure, helping him decide how much more testing may be warranted.

WYSIWYT/ML also allows the assistant to leverage Adam's positive tests (his "right" and "maybe right" marks) as training data—an extra benefit for Adam. (Only Adam's explicit tests become training data, not WYSIWYT/ML's inferred tests.) As previously mentioned, however, collecting a few training instances in this way is not the point of WYSIWYT/ML. Our goal is to allow Adam to effectively and efficiently assess how much he can rely on the assistant, not to collect enough training instances to fix its flaws.

Use-Case UC-2: Adam: "It was reliable before; is it reliable now?" Adam's intelligent assistant continually learns from Adam's behaviors, changing its reasoning based upon Adam's feedback. The assistant may also encounter data unlike any it had seen before. Hence, for use-case UC-2, WYSIWYT/ML's fourth functionality is to *monitor* coverage over time, alerting Adam when a previously tested assistant is exposing behaviors that he has not yet assessed.



Subject	Date ▲	Topic	Confidence	Correctness	History
Re: insect impacts	Apr 6, 07:45 AM	Motorcycles		✓	
Re: Kawi Zephyr? (was R...	Apr 6, 08:35 AM	Cars		✖	
OTTOMENU ... Where Ca...	Apr 6, 09:02 AM	Computers	●	✓	● ?
Version control for MAC a...	Apr 6, 11:07 AM	Computers		✓	

✓ 16% | ✖ | ? 78%

Fig. 4. (Top): The user tested three of the messages (the dark check marks and X marks), so they no longer show a priority. Then the computer inferred the third message to be correct (light gray check mark). Because the user's last test caused the computer to infer new information, the *History* column shows the prior values of what changed. (These values move right with each new change, until they are pushed off the screen.) (Bottom): A *test coverage bar* informs users how many topic predictions have been judged (by the user or the computer) to be correct (check mark) or incorrect (X mark).

Whenever Adam tests one of the assistant's predictions or new content arrives for the assistant to classify, the system immediately updates all of its information. This includes the assistant's predictions (except for those Adam "locked down" by explicitly approving them), all testing priorities, all inferred tests, and the test coverage bar. Thus, Adam can always see how "tested" the assistant is at any given moment. If he decides that more testing is warranted, he can quickly tell which predictions WYSIWYT/ML thinks are the weakest (UC-1) and which predictions are not covered by his prior tests (UC-2).

4.2 Cognitive Dimensions Analysis

We used Cognitive Dimensions [11], a popular analytical technique, to head off problems early in the design of our WYSIWYT/ML prototype. This analysis revealed four key issues for the implementation of WYSIWYT/ML, which we addressed as follows.

What just changed and how (Hard Mental Operations, Hidden Dependencies). Hard mental operations denote the user having to manually track or compute things in their head, and a hidden dependency is a link between two items that is not explicit. These dimensions revealed that a user could only answer the question "what just changed?" by scrolling extensively and memorizing prior statuses. To solve this, we added a History column showing the last two statuses of each message.

Too eager to help (Premature Commitment). This dimension denotes requiring users to make a decision before they have information about the decision's consequences. Because each user action may cause the assistant to update its predictions and WYSIWYT/ML to update its priority rankings and inferred tests, end user cannot easily guess the scope of alterations resulting from each interaction. In an early prototype, testing a prediction could cause on-screen messages to disappear from the user's view (due to the system automatically re-sorting messages based on new predictions or test priorities), making it difficult to see these consequences. Thus, we changed the prototype to only re-sort when the user asks it to (e.g., clicks the column header). This modification also helps emphasize recent changes, as other affected items in the sort "key" become visually distinct from their neighbors (e.g., now have a lower test priority than nearby messages).

Notes and scratches (Secondary Notation). Secondary notations allow users to annotate, change layout, etc., to communicate informally with themselves or with other humans in their environment (as versus communicating with the computer). We decided that secondary notation was unnecessary for end-user testing. As our empirical results will show, revisiting this decision may be warranted—some participants appeared to repurpose certain user interface affordances to indicate assistant predictions they intended to revisit later.

Communication overload (Role Expressiveness). This dimension denotes a user's ability to see how a component relates to the whole. This was initially a problem for our priority widget because it had too many roles: a single widget communicated the *priority* of assessing the message, explained *why* it had that priority, and *how* the message had been assessed—all in one small icon. Thus, we changed the prototype so that no widget had more than one role. We added the Correctness column to show the user's (or computer's) assessment (Figure 1), the green square to represent priority, and the widgets inside to explain the reasoning behind the priority (Figure 2).

5 Empirical Study

We conducted a user study to investigate use-case UC-1, the user's initial assessment of an assistant doing important work. We attempted to answer three research questions to reveal how well ordinary end users could assess their assistants in this use-case, even if they invested only 10 minutes of effort:

RQ1 (Efficacy): Will end users, testing systematically with WYSIWYT/ML, find more bugs than via ad hoc testing?

RQ2 (Satisfaction): What are the users' attitudes toward systematic testing as compared to ad-hoc testing?

RQ3 (Efficiency): Will WYSIWYT/ML's coverage contributions to the partnership help with end users' efficiency?

We used three systematic testing treatments, one for each prioritization method (Confidence, Similarity, and Relevance). We also included a fourth treatment (Control) to represent ad hoc testing. Participants in all treatments could test (via check marks, X marks, and label changes) and sort messages by any column in the prototype. See Figure 1 for a screenshot of the Confidence prototype; Similarity and Relevance looked similar, save for their respective prioritization methods and visualizations (Figure 2). Control supported the same testing and sorting actions, but lacked prioritization visualizations or inferred tests, and thus did not need priority/inferred test history columns.

The experiment design was within-subject (i.e. all participants experienced all treatments). We randomly selected 48 participants (23 males and 25 females) from respondents to a university-wide request. None of our participants were Computer Science majors, nor had any taken Computer Science classes beyond the introductory course. Participants worked with messages from four newsgroups of the widely used 20 Newsgroups dataset [16]: *cars*, *motorcycles, computers,* and *religion* (the original rec.autos, rec.motorcycles, comp.os.ms-windows.misc, and soc.religion.christian newsgroups, respectively). This data set provides real-world text for classification, the performance of machine learning algorithms on it is well understood, and, most important, the "gold standard" topic choice (the newsgroup to which the message's author posted it) defines exactly which messages are "bugs" (misclassified by the assistant), in turn defining how many of those bugs participants found and when WYSIWYT/ML's inferred approvals went astray.

We randomly selected 120 messages (30 per topic) to train the intelligent assistant using a support vector machine [6]. We randomly selected a further 1,000 messages over a variety of dates (250 per topic) and divided them into five data sets: one tutorial set (to familiarize our participants) and four *test sets* (to use in the main tasks). Our intelligent assistant was 85% accurate when initially classifying each of these sets. We used a Latin Square to counterbalance treatment orderings and randomized how each participant's test data sets were assigned to the treatments.

Participants answered a background questionnaire, then took a tutorial to learn one prototype's user interface and to experience the kinds of messages and topics they would be seeing during the study. Using the tutorial set, participants practiced testing and finding the assistant's mistakes in that prototype. For the first main task, participants used the prototype to test and look for mistakes in a 200-message test set. After each treatment, participants filled out a Likert-scale questionnaire with their opinions of their success, the task difficulty, and the prototype. They then took another brief tutorial explaining the changes in the next prototype, practiced, and performed the main task in the next assigned data set and treatment. Finally, participants answered a questionnaire covering their overall opinions of the four prototypes and comprehension.

Table 1. ANOVA contrast results (against Control) by treatment. The highest values in each row are shaded.

	Mean (*p*-value for contrast with Control)				df	F	*p*
	Confidence	Similarity	Relevance	Control			
Bugs Found (max 30)	12.2 (*p*<.001)	10.3 (*p*<.001)	10.0 (*p*<.001)	6.5 (N/A)	3, 186	10.61	<.001
Helpfulness (max 7)	5.3 (*p*<.001)	5.0 (*p*<.001)	4.6 (*p*<.001)	2.9 (N/A)	3, 186	22.88	<.001
Perceived Success (max 21)	13.4 (*p*=.016)	13.3(*p*=.024)	14.0 (*p*=.002)	11.4 (N/A)	3, 186	3.82	.011

6 Results

6.1 RQ1 (Efficacy): Finding Bugs

Bugs Found. To investigate how well participants managed to find an assistant's mistakes using WYSIWYT/ML, we compared bugs they found using the WYSIWYT/ML treatments to bugs they found with the Control treatment. An ANOVA contrast against Control showed a significant difference between treatment means (Table 1). For example, participants found nearly twice as many bugs using the frontrunner, Confidence, than using the Control version.

Not only did participants find more bugs with WYSIWYT/ML, the more tests participants performed using WYSIWYT/ML, the more bugs they found (linear regression, F(1,46)=14.34, R^2=.24, beta=0.08, *p*<.001), a relationship for which there was no evidence in the Control variant (linear regression, F(1,45)=1.56, R^2=.03, beta=0.03, *p*=.218). Systematic testing using WYSIWYT/ML yielded significantly better results for finding bugs than ad-hoc testing.

Profile of a Hard Bug. Our formative offline oracle experiments revealed types of bugs that would be hard for some of our methods to target as high-priority tests. (Recall that, offline, Relevance and Similarity were better than Confidence in this respect.) In order to evaluate our methods with real users, we took a close look at Bug 20635, which was one of the hardest bugs for our participants to find (one of the five least frequently identified). The message topic should have been Religion but was instead predicted to be Computers, perhaps in part because Bug 20635's message was very short and required domain-specific information to understand (which was also true of the four other hardest bugs):

> *Subject: Mission Aviation Fellowship*
> *Hi, Does anyone know anything about this group and what they do? Any info would be appreciated. Thanks!*

As Table 2 shows, nearly all participants who had this bug in their test set found it with the Relevance treatment, but a much lower fraction found it using the other treatments. As the table's Prioritization column shows, Relevance ranked the message as very high priority because it did not contain any useful words, unlike Confidence (the assistant was very confident in its prediction), and unlike Similarity (the message was fairly similar to other messages). Given this complementarity among the different methods, we hope in the future to evaluate a combination (e.g., a weighted average or voting scheme) of prioritization methods, thus enabling users to quickly find a wider variety of bugs than they could using any one method alone.

6.2 RQ2 (Satisfaction): User Attitudes

Participants appeared to recognize the benefits of systematic testing, indicating increased satisfaction over ad hoc testing. When asked "*How much did each system help you find the computer's mistakes?*" on a seven-point Likert scale, an ANOVA contrast again confirmed that responses differed between treatments (Table 1, row 2), with WYSIWYT/ML treatments rated more helpful than Control. Table 1's 3rd row shows that participant responses to the NASA-TLX questionnaire triangulate this result. Together, these results are encouraging from the perspective of the Attention Investment Model—they suggest that end users can be apprised of the benefits (so as to accurately weigh the costs) of testing an assistant that does work important to them.

Table 2. The number of participants who found Bug 20635 while working with each WYSIWYT/ML treatment

Treatment	Prioritization	Found	Did not find
Confidence	0.14	9	15
Similarity	0.58	11	14
Relevance	1.00	19	4

6.3 RQ3 (Efficiency): The Partnership's Test Coverage

Recall that when a participant tested a message, the system partnered with the user by inferring additional tests to "cover" similar messages (recall Figure 4). Coverage can be a powerful concept: it enables a user to reduce the number of items they must look over while still gaining a reasonable understanding of the assistant's reliability. It also reveals the weaknesses of an assistant's reasoning in terms of areas not yet covered by tests. In other domains, research has generally found that increased coverage increases bug finding [5, 8, 25]. Thus, in this section, we consider how much coverage the partnership achieved and how this related to participants' efficiency.

Table 3. Tests via check marks, X marks, and topic changes during a 10-minute session (out of 200 total messages per session), for the three WYSIWYT/ML treatments

	Mean √s participants entered per session	Mean Xs participants entered per session	Mean √s inferred per session	Mean Xs inferred per session	Total √s	Total Xs
Explicit	Regular: 35.0 "Maybe": 7.1	Regular: 2.4 "Maybe": 2.7	Regular: 46.4 "Maybe": 8.5	Regular: 4.7 "Maybe": 2.2	105.2	20.2
Implicit	8.2 topic changes as shortcuts for X+topic+√		N/A[1]			
Total tests	50.3	13.3	54.9	6.9		
Total messages tested [2]					117.2	

[1] Although the computer sometimes did change topics, this was due to leveraging tests as increased training on message classification. Thus, because these topic changes were not directly due to the coverage (cosine-similarity) mechanism, we omit them from this coverage analysis.

[2] *Total Tests* is larger than *Messages Tested* because topic changes acted as two tests: an X on the original topic, then a √ on the new topic.

Coverage: How much? Using WYSIYWT/ML, our participants were able to leverage their explicit tests by a factor of about 2. Together with the computer-oracle-as-partner, participants' mean of 55 test actions using WYSIWYT/ML covered a mean of 117 (60%) of the messages—thus, participants gained 62 inferred tests "for free". Table 3 shows the raw counts. With the help of their computer partners, two participants even reached 100% test coverage, covering all 200 messages within their 10-minute time limit.

Further, coverage scaled well. In an offline experiment, we tried our participants' explicit tests on the *entire* set of Newsgroup messages from the dates and topics we had sampled for the experiment—a data set containing 1,448 messages. (These were tests participants explicitly entered using either WYSIWYT/ML or Control, a mean of 55 test actions per session.) Using participants' 55 explicit tests (mean), the computer inferred a mean of 568 tests per participant, for a total coverage of 623 tests (mean) from only 10 minutes of work—a 10-fold leveraging of the user's invested effort.

Participant and WYSIWYT/ML Approvals vs. Disapprovals. As Table 3 shows, participants approved more messages than they disapproved. When participants approved a message, their topic choice matched the 20-Newsgroup "gold standard" (the original newsgroup topic) for 94% of their regular checkmarks and 81% of their "maybe" checkmarks (the agreement level across both types of approval was 92%). By the same measure, WYSIWYT/ML's approvals were also very accurate, agreeing with the gold standard an average 92% of the time—exactly the same level as the participants'.

Participants' regular X marks agreed with the gold standard reasonably often (77%), but their "maybe" X marks agreed only 43% of the time. Informal pilot interviews revealed a possible explanation: re-appropriation of the "maybe" X marks for a subtly different purpose. When unsure of the right topic, pilot participants said they marked it as "maybe wrong" to denote that it *could* be wrong, but with the intention to revisit it later. This indicates that secondary notation (in addition to testing notation)—in the form of a "reminder" to revisit instead of a disapproval—could prove useful in future prototypes.

Perhaps in part for this reason, WYSIWYT/ML did not correctly infer many bugs—only 19% of its X marks agreed with the gold standard. (The computer's regular X marks and "maybe" X marks did not differ—both were in low agreement with the gold standard.) The problem cannot be fully explained by participants repurposing "maybe" X marks—WYSIWYT/ML's regular inferred X marks were just as faulty. However, this problem's impact was limited because inferred X marks only serve to highlight possible bugs. Thus, the 81% failure rate on WYSIWYT/ML's average of seven X's per session meant that participants only had to look at an extra five messages/session. Most inferred tests were the very accurate *positive* tests (average of 55 per session), which were so accurate, participants could safely skip them when looking for bugs.

7 Discussion

Will end users *really* explicitly and systematically test an intelligent assistant? Although we did not test this question in our lab study, theory suggests that they will when the perceived benefits of doing so outweigh the costs [4]. Until this question can be investigated empirically, we target the subset of end users who are willing to expend at least modest effort to assess assistants on tasks in which mistake types and frequencies *must* be understood before the user would be willing to rely on them, such as with Adam's intelligent qualitative coding assistant.

Our current similarity-based notion of coverage also warrants further empirical investigation. It worked well for approvals, but a smaller threshold for disapprovals may result in fewer false bug identifications. In the future, we plan a systematic evaluation of this threshold and its impact on different aspects of WYSIWYT/ML.

Finally, we emphasize that finding (not fixing) bugs is WYSIWYT/ML's primary contribution toward debugging. Although WYSIWYT/ML leverages user tests as additional training data, simply adding training data is not an efficient method for *debugging* intelligent assistants. To illustrate, our participants' *testing* labeled, on average, 55 *messages,* which increased average accuracy by 3%. In contrast, participants in another study that also used a subset of the 20 Newsgroup dataset spent their 10 minutes *debugging* by specifying *words/phrases* associated with a label [32]. They entered only about 32 words/phrases but averaged almost twice as much of an accuracy increase (5%) in their 10 minutes. Other researchers have similarly reported that allowing users to debug by labeling a word/phrase is up five times more efficient than simply labeling training messages [23]. Thus, rather than attempting to replace the interactive debugging approaches emerging for intelligent assistants (e.g., [17, 18, 22, 30]), WYSIWYT/ML's bug-finding complements them. It provides the missing testing piece, suggesting where important bugs have emerged and when those bugs have been eradicated, so that end users need not debug blindly.

8 Conclusion

With the increase in intelligent assistants helping with critical tasks comes the need to rethink the nature of how end users can assess whether and when to rely on their assistants' help. WYSIWYT/ML is the first work to address this need.

WYSIWYT/ML is a human/computer partnership that enables end users to assess intelligent assistants systematically. The human's role is to approve or disapprove (i.e., test) portions of the assistant's work. The computer's role is to *advise* the user about testing priorities, *contribute* additional tests similar to the user's (which the user may verify), *measure* how much of the assistant's reasoning has been assessed, and *monitor* the need for additional assessment as the assistant evolves over time.

Our empirical evaluation showed that systematically testing with WYSIWYT/ML resulted in a significant improvement over ad hoc methods in end users' abilities to assess their assistants: our participants found almost twice as many bugs with our best WYSIWYT/ML variant as they did while testing ad hoc. Further, the approach scales: participants covered 117 messages in the 200-message data set (over twice as many as they explicitly tested) and 623 messages in the 1448-message data set (over 10 times as many as they explicitly tested)—all at a cost of only 10 minutes work.

Thus, systematic assessment of intelligent assistants was not only effective at finding bugs—it also helped ordinary end users assess a reasonable fraction of an assistant's work in a matter of minutes. These findings strongly support the viability of bringing systematic testing to this domain, empowering end users to judge whether and when to rely on intelligent assistants that support critical tasks.

Acknowledgements. We thank Jeremy Goodrich, Travis Moore, Shalini Shamasunder, Nicole Usselman, Chaoqiang Zhang, and our study participants for their help. This work has been supported in part by NSF 0803487.

References

[1] Abraham, R., Erwig, M.: AutoTest: A tool for automatic test case generation in spreadsheets. In: Proc. VL/HCC, pp. 43–50. IEEE, Los Alamitos (2006)

[2] Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston (1999)

[3] Beizer, B.: Software Testing Techniques. International Thomson Computer Press (1990)

[4] Blackwell, A.: First steps in programming: A rationale for attention investment models. In: Proc. HCC, pp. 2–10. IEEE, Los Alamitos (2002)

[5] Burnett, M., Cook, C., Rothermel, G.: End-user software engineering. Comm. ACM 47(9), 53–58 (2004)

[6] Chang, C., Lin, C.: LIBSVM: A library for support vector machines (2001), `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

[7] Fisher, M., Cao, M., Rothermel, G., Brown, D., Cook, C., Burnett, M.: Integrating automated test generation into the WYSIWYT spreadsheet testing methodology. ACM Trans. Software Engineering and Methodology 15(2), 150–194 (2006)

[8] Frankl, P., Weiss, S.: An experimental comparison of the effectiveness of branch testing and data flow testing. IEEE Trans. Software Eng. 19(3), 202–213 (1993)

[9] Glass, A., McGuinness, D., Wolverton, M.: Toward establishing trust in adaptive agents. In: Proc. IUI, pp. 227–236. ACM, New York (2008)

[10] Gmail Priority Inbox: Get through your email faster, `http://google.com/mail/help/priority-inbox.html` (accessed September 16, 2010)

[11] Green, T., Petre, M.: Usability analysis of visual programming environments: A cognitive dimensions framework. J. Visual Languages and Computing 7(2) (June 1996)

[12] Grigoreanu, V., Cao, J., Kulesza, T., Bogart, C., Rector, K., Burnett, M., Wiedenbeck, S.: Can feature design reduce the gender gap in end-user software development environments? In: Proc. VL/HCC, pp. 149–156. IEEE, Los Alamitos (2008)

[13] Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer, Heidelberg (2003)

[14] IEEE, IEEE Standard Glossary of Software Engineering Terminology (IEEE Std610.12-1990) (1990)

[15] Klann, M., Paterno, F., Wulf, V.: Future perspectives in end-user development. In: Lieberman, H., Paterno, F., Wulf, V. (eds.) End-User Development. Springer, Heidelberg (2006)

[16] Kniesel, G., Rho, T.: Newsgroup data set (2005), http://www.ai.mit.edu/jrennie/20newsgroups

[17] Kulesza, T., Wong, W., Stumpf, S., Perona, S., White, R., Burnett, M., Oberst, I., Ko, A.: Fixing the program my computer learned: Barriers for end users, challenges for the machine. In: Proc. IUI, pp. 187–196. ACM, New York (2009)

[18] Kulesza, T., Stumpf, S., Burnett, M., Wong, W., Riche, Y., Moore, T., Oberst, I., Shinsel, A., McIntosh, K.: Explanatory debugging: Supporting end-user debugging of machine-learned programs. In: Proc. VL/HCC. IEEE, Los Alamitos (2010)

[19] Lawrance, J., Bogart, C., Burnett, M., Bellamy, R., Rector, K., Fleming, S.: How programmers debug, revisited: An information foraging theory perspective. IEEE Trans. Software Engineering (2011)

[20] Lim, B., Dey, A., Avrahami, D.: Why and why not explanations improve the intelligibility of context-aware intelligent systems. In: Proc. CHI, pp. 2119–2128. ACM, New York (2009)

[21] Lim, B., Dey, A.: Toolkit to support intelligibility in context-aware applications. In: Proc. Int. Conf. Ubiquitous Computing. ACM, New York (2010)

[22] Miller, R., Myers, B.: Outlier finding: Focusing user attention on possible errors. In: Proc. UIST, pp. 81–90. ACM, New York (2001)

[23] Raghavan, H., Madani, O., Jones, R.: Active learning with feedback on both features and instances. JMLR 7, 1655–1686 (2006)

[24] Raz, O., Koopman, P., Shaw, M.: Semantic anomaly detection in online data sources. In: Proc. ICSE, pp. 302–312. IEEE, Los Alamitos (2002)

[25] Rothermel, G., Burnett, M., Li, L., Dupuis, C., Sheretov, A.: A methodology for testing spreadsheets. ACM Trans. Software Engineering and Methodology 10(1) (January 2001)

[26] Rowan, J., Mynatt, E.: Digital family portrait field trial: Support for aging in place. In: Proc. CHI, pp. 521–530. ACM, New York (2005)

[27] Scaffidi, C.: Unsupervised inference of data formats in human-readable notation. In: Proc. Int. Conf. Enterprise Integration Systems, pp. 236–241 (2007)

[28] Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison (2009)

[29] Shen, J., Dietterich, T.: Active EM to reduce noise in activity recognition. In: Proc. IUI, pp. 132–140. ACM, New York (2007)

[30] Talbot, J., Lee, B., Kapoor, A., Tan, D.: EnsembleMatrix: Interactive visualization to support machine learning with multiple classifiers. In: Proc. CHI, pp. 1283–1292. ACM, New York (2009)

[31] Tullio, J., Dey, A., Chalecki, J., Fogarty, J.: How it works: A field study of non-technical users interacting with an intelligent system. In: Proc. CHI, pp. 31–40. ACM, New York (2007)

[32] Wong, W.-K., Oberst, I., Das, S., Moore, T., Stumpf, S., McIntosh, K., Burnett, M.: End-user feature labeling: A locally-weighted regression approach. In: Proc IUI. ACM, New York (2011)

An End-User Oriented Building Pattern for Interactive Art Guides

Augusto Celentano and Marek Maurizio

Università Ca' Foscari Venezia
{auce,marek}@dsi.unive.it

Abstract. A primary issue in the design of interactive art guides is the effort to build them as a synthesis of several cultures and skills concerning content editing, multimedia production, application structuring and interaction design. The communication between the actors of the development process can be boosted by involving the cultural domain experts in the whole life cycle, providing them with proper design environments for extending and modifying the guide structure and content, to refine the initial design without the intervention of computer specialists. We have developed a framework for End-User Development (EUD) of interactive multimedia guides based on experiences in art exhibitions held at Ca' Foscari in the last two years, culminating in the project of a multimedia guide for the François Pinault contemporary art collection at *Punta della Dogana* in Venice. The guide features dynamically generated tours personalized on the user answers to a set of questions asked during the tour, prepared by a domain expert as part of the guide content. In this paper we present a methodology for designing end-user oriented software environments based on open and portable standards, and discuss the development of a content management systems for domain experts able to generate personalized tours.

Keywords: Building pattern, end-user development, interactive multimedia guides, web application.

1 Introduction

Multimedia interactive guides are replacing at fast pace the old-fashioned audio guides in city tours, museums and temporary exhibitions. The development of powerful mobile devices, fast wireless communication and rich interaction styles boost the production of mobile applications devoted to tourism and art, available not only on-site but also in large on-line markets.

Pioneering experiments with devices based on Microsoft Windows Mobile and Apple iOS have been made in the last decade by prominent cultural institutions like the New York Moma, the Tate Liverpool Gallery, the Barcelona Ship Museum, the San Jose Museum of Art, to cite a few [1,2,3,4]. Web sites and portals of museum offered mobile-specific sections with the possibility of downloading multimedia content. The Apple AppStore and Android Market are changing the landscape of applications available to tourists and museum visitors, which are delivered not only as a support to on-site tours, but also (and sometimes primarily) as an independent product of the issuing institution "brand". The AppStore returns hundreds of titles as answers to the queries "art

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 187–202, 2011.

guide" and "museum guide". Many of them are electronic versions of traditional touristic guides, and the quality is sometimes questionable, but the growth of interest in this market is evident.

Among the most interesting examples we cite the MoMA iOS application [1] that allows users to access information about the artworks of the New York MoMA collection, informs users on current and future events, and lets them to access a series of tours. The application is available also for Android devices. Other similar applications have been developed in Europe by Versailles Gardens, with an interesting view of the castle's garden by proposing discovery paths; by Tate Trumps in UK, with games and puzzles to learn about the Tate Modern Gallery artworks; by the Vatican Museum of Rome, with a detailed view on some of the most important masterpieces of Italian art. Temporary exhibitions complete the scenario with guides increasingly based on interactive multimedia rather than on bare audio; a balance between the greater effort to produce them and the improved attractiveness for the public is leaning towards the new technologies.

Critical issues in the design of mobile guides are the choice of the amount, detail and style of information, the design of the proper access and navigation paths and the adaptation of a potentially huge set of multimedia material to a wide spread of visitors; as a consequence, new methods and tools to design, implement and evaluate the new applications are needed and, indeed, are emerging [8,15,16]. We argue that, mainly when such applications are conceived as the outcome of interdisciplinary research environments, it is important to devise not only a set of final products, but also a set of frameworks where programmers, designers and domain experts can cooperate in the development and extension of the software artifacts. Following this principle, the Indianapolis Museum of Art released *TAP*, an open-source tool to create and deliver mobile tour applications in a museum setting [9]. The *TAP* system is based on a content management system that exports data into an intermediate format, *TourML*, which can then be used as a pluggable bundle for mobile applications.

In this paper we describe the *Punta della Dogana* project (hereafter referred to as *PuD*), an academic research project aiming to set up a modern critical instrument to understand contemporary art with a double focus: at one hand, to provide visitors with comprehensive information about an art collection, an institution, a set of artists, with rich multimedia material prepared by experts in the domain: researchers, art critics, collectors, etc.. At the other hand, to provide not only designers and programmers but also domain experts (e.g., art critics) with a system in which they can gather, organize and tailor multimedia content of heterogeneous nature, instantiating a set of predefined templates on actual "visit paths" for a wide range of users of mobile devices. The project outcomes are, at the first stage, a Content Management System (CMS) oriented to domain experts and a mobile guide on Apple iOS devices with comprehensive information about a sample of the art collection of the François Pinault Foundation at Punta della Dogana in Venice.

We have developed a structure for multimedia art guides that departs from the classical menu based application and is, instead, based on a directed graph of visits built on top of selected topics. We argue that such approach, if paired with an accurate selection of multimedia content by domain-experts, leads to an interesting and new way of presenting contents to art visitor. In particular, in situations where a large set of multimedia items is available, our model allows domain experts to categorize it and lets the user to choose, more or less transparently, a personal selection to experience.

The *PuD* project takes inspiration from our previous work with multimedia installations and with mobile guides on Apple iOS devices, designed for several art exhibition held at Ca' Foscari in Venice in 2009 and 2010 [5,6]. The *PuD* project is the culmination of those experiences: as an improvement with respect to the past projects, it aims to give the visitor the ability to create a personalized tour through a sequence of questions that progressively builds the tour's steps based not only on the visitor's background and general interests but also on the attention and curiosity developed during the tour itself. The development system to be used by domain experts to define the guide structure is based on a multimedia repository on which multiple perspectives, multiple navigation paths and multiple interaction styles can be adapted and dynamically composed, fulfilling the requirements of different categories of users. The resulting experience is thus built upon a spread of cultural and emotional feedbacks coming from the user background and also from the reaction to the actual exhibition contents.

In developing the CMS and the *PuD* mobile guide we followed the end user development principles presented in [7,10] and produced a framework where programmers, interface designers, and domain experts can cooperate in the development and extension of the software artifact. We built our framework to be as much re-usable as possible for future projects with similar characteristics. Research in this area usually seeks to develop domain-specific or graphic modeling languages that allow users to easily express the desired functionalities [14]: we initially used a simple text-based *domain-specific language*, but later we also developed a web-based environment to shorten the conceptual gap between the technical view of programmers and the more abstract view required by the end-user development principles [12].

2 The PuD Mobile Guide

Even a well informed public may find difficult to access and to understand contemporary art. Contemporary art is not yet historicized and in continuous development; it lacks a consolidated criticism and escapes fixed classification schemes, resulting in a very complex field of investigation. We believe that a way to develop knowledge in contemporary art is to make appealing to explore, with a user-friendly approach, its different contexts to understand how they contribute to build its cultural value; the possibilities offered by new devices to interact with engaging audiovisual content can attract the user much more than simple traditional, often boring, audio narrations.

The *PuD* project focuses on a selection of artists and artworks from the collection of the François Pinault Foundation in the exhibition *Mapping the Studio* at Punta della Dogana in Venice. The *PuD* project retraces the course of each art piece, from its conception in the studio up to its integration within the private collection, analyzing the artist's conception of art and the social and historical events occurring during the artist's work.

The experts in contemporary art that are following this project have selected a small group of artists: Jake & Dinos Chapman, Takashi Murakami, Rachel Whiteread, Huang Yong Ping and Rob Pruitt. They are representative of the Punta della Dogana collection, and showed much enthusiasm for collaborating to the project; moreover, some of their artworks are highly suitable for a multimedia interpretation, because rich of miniaturized figures like Jake & Dinos Chapman's *Fucking Hell* and Huang Yong Ping's *Football Match*; their reading could be facilitated by a close visual investigation, not possible on the original artworks.

The art experts collected the basic critical documentation, mainly audio-video resources. They interviewed the artists to build a consistent picture of each artist and artwork to build the guide skeleton. The interviews consisted in a series of questions based on a standard pattern touching similar subjects for all cases, to anticipate the possibility of comparing them during the guide exploitation: a presentation of the artist made by him/herself; reasons for deciding to be an artist; a personal view of the role of the artist in contemporary society; a description of the studio and of the artistic creation process; a commentary about the specific artwork; general comments about own's work; finally, the relationship with the Pinault Collection and with the city of Venice. More specific themes were addressed to consider the artist's personal history and ideas. The collected material was completed with commentaries of the exhibition curators and a sample from a lecture series about the Punta della Dogana collection organized by the Venice Universities in 2010.

A complementary section of the project is devoted to the architecture of the Punta della Dogana building, from its origins until the recent restoration by the Japanese architect Tadao Ando, and to a selection of interesting viewpoints used in the guide as environments for general discussions. The collection of the material produced a very large set of heterogeneous files, that were processed and organized to build a multimedia repository, a project database on which experts, interface designers and application programmers could work together to progressively refine the mobile guide content.

To avoid a cognitive overload on the visitor with so much material we based the guide conceptual architecture on a structured set of questions, multiple choice criteria that allow the visitors to enjoy a dynamic tour based on their own interaction with the system, making them active users building their own interactive journey in the exhibition [11].

2.1 The PuD Guide Model

The guide structure is defined by an abstract model based on the notion of *topic* and *visit*. A topic represents a subject of interest in the exhibition: it can be abstract, like an historical period, or concrete, like a painting. In the *PuD* project, for instance, the topics are five artworks and four architectural points of interest. Each topic of the guide is composed by a set of related contents. A visit, instead, represent the path a user experiences through the set of proposed contents about a specific topic. The structure of each topic is described by a *deterministic finite automaton* $M = (Q, \Sigma, \delta, q_0, F)$ where:

- Q is a finite set of states; each state is the abstract representation of a content;
- Σ is a finite set of inputs, which are atomic user interactions like pressing a button, selecting an image, tilting the device, etc.;
- δ is a transition function $(Q \times \Sigma \rightarrow Q)$;
- q_0 is the starting state, marking the beginning of a visit;
- F is a set of final states, each representing a possible end of a visit.

In state q_0 a summary of the topic is presented to the visitor (the artwork's title, the author's name, etc.). The remaining states $Q - q_0$ are partitioned in two disjoint subsets Q_c and Q_q, called *contents* and *questions*, respectively. When the guide is in a content state the visitor is presented a node containing a multimedia piece of information. Each content state is associated with only one transition: after experiencing the content, the visitor goes to the next state without any explicit interaction. Question states, instead, present the users a node containing two or more alternatives to choose from: the visitor interaction (the answer to the questions) is used to select the next state, i.e., the next node shown. When the system reaches a final state in F the visit ends: the guide presents the user a summary of the experienced contents and allows him/her to select a new topic.

A visit is therefore the sequence of states $q_0, q_1, \ldots, q_n$ experienced by the visitor. Visits are used immediately to give the user a summary of the experienced contents, but are also stored persistently on the device for subsequent analysis of the paths followed: our experience with mobile guides proved that such analysis is a valuable evaluation tool for tuning the guide design [6].

2.2 The PuD Guide Implementation

According to [10] the goal of human computer interaction is evolving from just making systems that are easy to use to systems that are easy to develop. This is of particular importance in the field of computer science applied to cultural heritage, where different areas of expertise require different approaches, and often data is biased by some degree of subjectivity. For this reason we departed from the simple solution of developing an ad-hoc structure for our contents, but produced instead a framework where domain experts can participate in the development of the guide.

The navigational structure of the guide is quite trivial, composed by a set of pages and menus. We focused our development process on the creation and presentation of contents and questions. A guide page corresponds to a node of the abstract model, and is generated starting from a tuple (T, D, St, Sc, F), where:

- T is an HTML document that contains the node's markup. This document works as a template and defines the general structure of the node.
- D is a document containing the node's data (actually, a JSON text document).
- St is a CSS document containing styles to apply to the node markup.
- Sc is a Javascript document to be executed after the node has loaded.
- F is a set of external files that can be linked by the node, like images, videos, and so on.

When a specific node is requested the application fetches the required files and combines them to create the actual content that is presented to the user.

The design of such a product, however, tends to become fairly quickly outdated or insufficient because of changing requirements. The conventional view of *design before use* is challenged by a new approach of *design during use* [12,13]. Our system has been designed to be *adaptable* by end-users of different classes and skills, following the principle of *gentle slope*: modifications and personalizations can be carried out at different

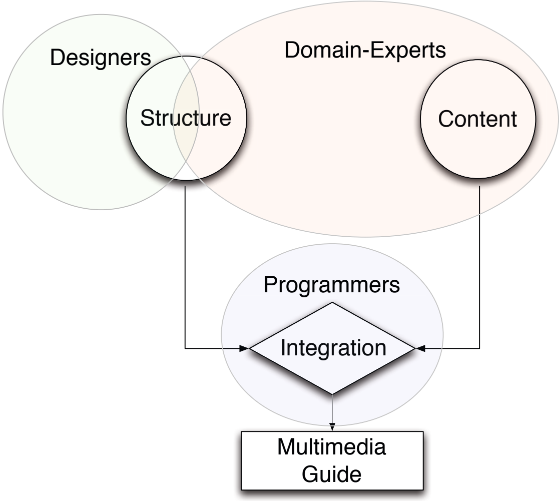


Fig. 1. The guide development architecture

levels of increasing complexity. Our architecture allows us to extend and modify the guide at three different levels:

- at the lower and most complex level a *programmer* can act on the application source code to modify or extend the basic behavior of the guide: this level requires a deep knowledge of the Javascript programming language, Ajax calls, and knowledge of the development framework, but allows also the programmer to include completely new functionalities, like, for instance, the possibility of controlling a shared, large screen with the portable device;
- at the middle level a *designer* can act on T, St, and Sc to create new node templates: this level require knowledge of HTML, CSS and, optionally, some Javascript to attach dynamic behavior to the new types.
- at the higher level a *domain expert* with little or no prior knowledge of HTML, CSS, or Javascript can act on D and F, the node data and the external multimedia files, to create content that will be applied to the pre-existing templates to generate new node instances. This solution requires only the knowledge of some simple syntactical conventions to format the content and familiarity with the available node templates, which are part of the framework's documentation.

The system is thus organized according to three different perspectives corresponding to the development team roles, in three levels of increasing complexity. Domain experts can compose data files to insert new nodes and, defining relationships between them, creating new thematic, spatial, or logical organizations of artworks in the guide. There is no need of intervention by specialized actors to perform common tasks such as inserting data, organizing catalogs, defining multiple relationships between contents. Should the need of new, unforeseen, extensions arise, our building pattern takes into account two different levels of intervention: on a first level designers (actors with knowledge of web design technologies) can create new templates, different only in aesthetics or provide advanced behavior obtained through the Javascript programming language; on a second level programmers can extend the system by implementing new components.

Our modular approach, summarized in Figure 1, provides a gentle slope by allowing successive decomposition and reconfiguration of software entities that are themselves built from smaller components [7,10]. We produced a framework where different actors can cooperate in the development and extension of the software artifact, and the produced modules are re-usable as much as possible for new projects.

3 The Content Management System

The domain experts play a role in two cases: the design of the guide conceptual structure and the organization of the multimedia material from which the guide content is built. In the former case experts must work in strict cooperation with the interface designers and the programmers, to exchange feedback about templates and interaction styles. In the latter case they could be more independent, if provided with a set of software tools easy to use and oriented to their needs. General purpose databases and digital library systems provide rich classification functions but are too general and, often, scarcely suitable for handling large multimedia documents. Content management systems (CMS) for web sites and portals are targeted to web masters and need to be customized to be user friendly, but are a better alternative to design a usable system.

In our early experiences with interactive multimedia guides we managed data about artworks and multimedia files in ad-hoc ways, collecting them in a shared structure of files and folders suited to the application logical structure. While this approach has the advantage of immediacy and efficiency, it might introduce ambiguities and inconsistencies; dealing with large quantities of data can hinder the entire project in terms of development speed and error detection. For the multimedia guide designed for the exhibition *Nigra sum sed formosa; sacred and beauty of the christian Ethiopia* held at Ca' Foscari in 2009 we developed a very simple CMS whose purpose was primarily the organization of the artworks in several collections, to be used as indices in the guide to browse the exhibition content according to different perspectives: themes, rooms, artwork type. The CMS proved to be useful mainly in managing the changes needed to reflect late decisions by the curators about what objects to select and how to place them in the exhibition spaces.

In the PuD project, to archive, classify and query multimedia content we have developed our own CMS (a web application with a tailored interface), whose main goal is to boost cooperation and integration between domain experts and application designers; multimedia content is collected, archived and organized according to the guide conceptual structure defined by the domain experts, leaving them the possibility of attaching comments, tags, taxonomies, etc.; at the same time the application designers are free to access data in an efficient way to build the application. The multimedia files uploaded by domain experts are accessible not only through the CMS but also through a regular filesystem accessed by the mobile application. Names and structure of the files and folders are generated automatically by the application to avoid conflicts. This approach can be seen as a sort of low-level API: the possibility of accessing contents through a filesystem allows developers to use consistently also other applications, batch processing, backup procedures, and so on.

3.1 Overview of the CMS

The CMS is an online storage space with the following characteristics:

1. it is accessed through a web-based application, requiring neither a specific installation nor specialized plugins, with a graphical interface based on common knowledge and clear action feedback, allowing domain experts to easily operate it;
2. the CMS data model is basically hierarchical. This is a direct consequence of the initial constraint requiring all the uploaded contents to be finally accessible through a filesystem. The model can represent entities (authors, artists, or conferences), multimedia contents (video, audio, images or texts), and their relationships;
3. to enrich the classification system and to allow domain experts to trace non hierarchical relationships, the CMS introduces a set of vocabularies to organize contents among different categories. Vocabularies may represent taxonomies, i.e., catalogs of predefined terms, or folksonomies, i.e. catalogs dynamically built on free terms. Different content types can relate to different vocabularies. Such a flexible classifications allows the CMS users to have different views of the data, e.g., to visualize the items related to a certain topic independently from their location in a standard hierarchical schema.

When multimedia content is added to CMS, it has a double representation: an abstract representation and a concrete representation. The abstract representation can be accessed from the web application. The content can be explored by keyword, relations can be visualized in a graphical way, and so on. The abstract representation is useful for domain experts to access, categorize, and discuss the contents. The concrete representation, instead, is represented by the underlying filesystem generated by the CMS. The resulting structure can be handled without any need of external applications but, at the same time, is not polluted by subjectivity. The concrete representation is mainly useful to computer scientists that need a formal, self-describing, structure to develop complex software constructs with contents coming from disciplines their are not expert in.

3.2 The CMS Abstract Data Model

The CMS abstract data model has two different classes of objects: *entities* and *multimedia content*. An entity can be, for instance, an author, an artwork, or an event (e.g., a talk, a lecture). Multimedia content can be a text, a video, an audio or an image. Each object has a set of attributes (a name, a short description, a creation/modification date, etc.) and can be in relation with other objects. The types of relationships involved, however, are limited by the constraint of implementing a hierarchical, meaningful underlying filesystem. For this reason we designed the data structure based on assumptions shared with the domain experts. In the context of the *PuD* project the assumptions derive from the structure of a the visits; being based on guided tours, not on free exploration, the connections among data items can be classified in advance according to a few main perspectives.

The entities can be related as shown in Figure 2. Boxes are content types and arrows represent relations between types. Content types are further categorized in entities and multimedia contents. An arrow from a type A to a type B means that objects of type A

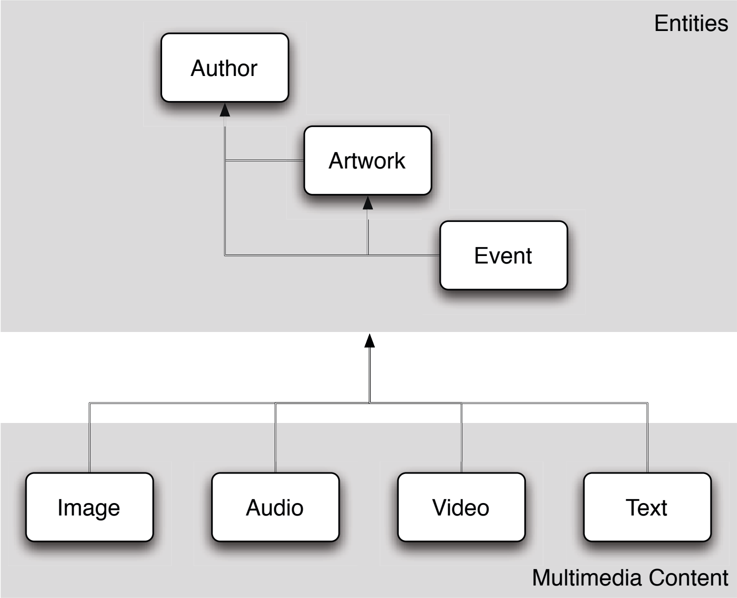


Fig. 2. The CMS abstract data model

maintain a reference to an object of type B. An event like a lecture, for instance, has a relation with either an artist (the one giving the lecture) or an artwork (the subject of the lecture). Each multimedia object can be in relation with any entity, either artists, artworks or events.

Multimedia contents are organized in groups: each group has a *master* element and a set of *derivates*. In a video object, for instance, the master element can be a whole interview with an artist, while the derivates can be meaningful clips of the same interview, technically refined and commented. In a text element, the master can be the transcript of a lecture and the derivates translations of the same text in different languages. The semantics of the master/derivate structure is left to the content authors by using description fields.

Images behave in a slight different way: since it's common to have a large number of images with the same subject without a specific master/derivate relation, we opted to drop this structure and to group images in simple collections.

3.3 Implementation

The CMS has been implemented in Drupal 6, a popular open-source content management system. We decided to use Drupal mainly for the time constraints given by the project: the development from scratch of a full-fledged content management system would have been too much time-consuming. Since the system must be used online by researchers without a computer science background it had to be secure and easy to use, with an intuitive interface. All these requirements could be met with a careful ad-hoc design, but not when developing from scratch in the short time required by our project. For this reason we found a good trade-off in adapting an already consolidated open source system to our needs.

The Drupal system allows users to create contents, also called *nodes*, based on predefined templates called *content types*. Each of these templates has a title, a description, and a set of typed *fields*. Fields can contain, for instance, text, dates, or files. Contents can be classified through vocabulary entries, both fixed terms or free tags, and can be related each other using a simple referencing mechanism. With these abstract blocks we built a web application where users can insert data about artists, artworks and conferences of the *PuD* project, classify them, add metadata, upload related multimedia files. The Drupal system features a fine-grained control on *permissions*: different classes of users can be allowed to edit, delete, comment, or view each type of content: all users, for instance, are allowed to view the contents and participate the discussions, while only editors can create or edit contents.

We are aware that using a general purpose software like Drupal can limit the innovation in the development process. Even if open source and easily extended, the Drupal system is still limited in its underlying structure. The major limitation, in our opinion, is the lack of an object-oriented design. The alternative, to be taken in consideration when writing a more general system, is to use an ad-hoc web application with a more complex data model based on object oriented principles. The physical filesystem representation does not have to be discarded, but it should not influence the data model: it should be possible to express a richer set of relationships, including many-to-many ones.

Another module of the CMS allows domain experts to browse between the different guide's node types and presents dynamic web forms to let users input the required data. Nodes are linked together to build the visit graph and the result is compiled in a *bundle*, a set of data and multimedia files, that can be used directly by the guide application.

4 A Case Study

In Figure 3 an example of an abstract topic graph is given: questions are represented by diamonds and contents are represented by circles. Each circle is labeled with a letter to denote the content type and a sequence number: *a* for audio, *t* for text, *i* for image, and *v* for video. Arrows represent the possible transitions between nodes. It is important to note that the topology of the graph, defined by domain experts, has a direct influence on how visitors can explore the exhibition with the assistance of the guide.

The graph in Figure 3 is the abstract representation of the topic related to the Rachel Whiteread's artwork *"Untitled"* (*One Hundred Spaces, 1995*). Figure 4 shows an instance of the abstract topic graph of Figure 3 with selected screenshots that represent a visit; each screenshot is labeled as the corresponding node in the abstract representation.

In state s and i_1 introductory contents are shown; in state q_1 the user is presented with a first visual question, asking to choose between information about the author, about the artwork, or about the atelier. When the user selects the first option another question q_2 is presented in text form, prompting the visitor to guess the motivation behind the artwork's title out of three possible answers. Figure 4 shows a specific point of view leading the visitor to a sequence of nodes about what the domain experts called

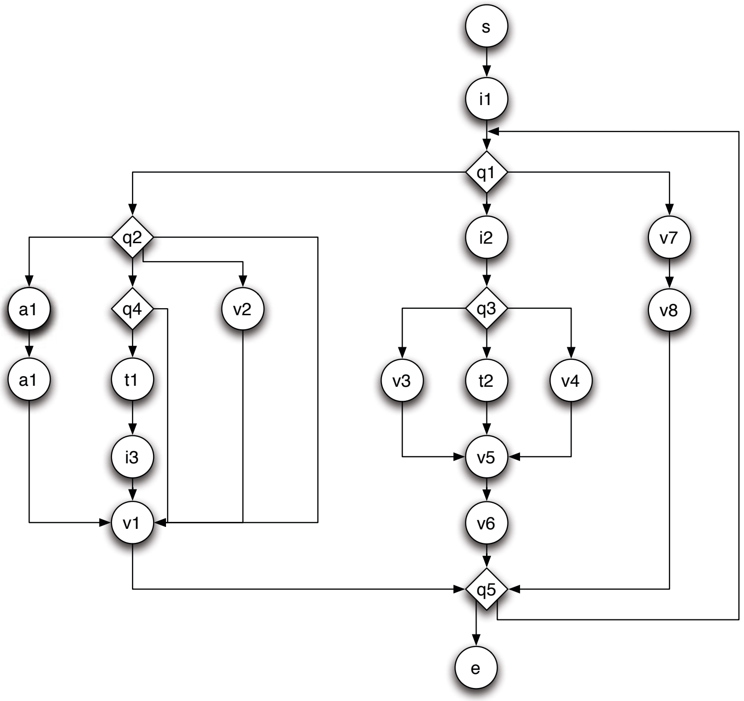


Fig. 3. Example of an abstract topic graph

"Building an invisible city". This part of the visit is characterized by a comparative approach to the artwork: the casts of chairs that compose the artist's installation look like the urban blocks of an imaginary city. Investigating the relationship between the sculpture and the architecture in Whiteread's artwork, the visitor is guided to explore this perspective. In state i_4 the concept is visually introduced by a graphic view of the City skyline in London behind the transparent surface of one of the artist's blocks; in state i_3 the user can read or listen to quotations of poetic and literary sources (e.g., excerpts from Italo Calvino's *Invisible cities*); in state v_9 the visitor can explore other Whiteread's installations where the concept of space is investigated in a progressive size reduction, from a city to a room. In the final state v_1, the user can listen to an excerpt of an interview where the artist directly describes and comments her work.

The guide subsequently asks the visitor if he/she wants to receive more information about this topic. A positive answer brings the user back to the artwork/artist/atelier selection, while a negative answer presents the visitor with a summary of the part of the visit just completed (represented by the chosen path of nodes) and let he/she to select another topic to continue. In the example of Figure 4 the visitor chooses to continue the visit with more information about the same artist. State i_2 features a brief introduction to the artist; in state q_3 three aspects of the artist's life are proposed, asking the visitor to select one: a spoken interview, where she speaks about herself, a biography (pictured

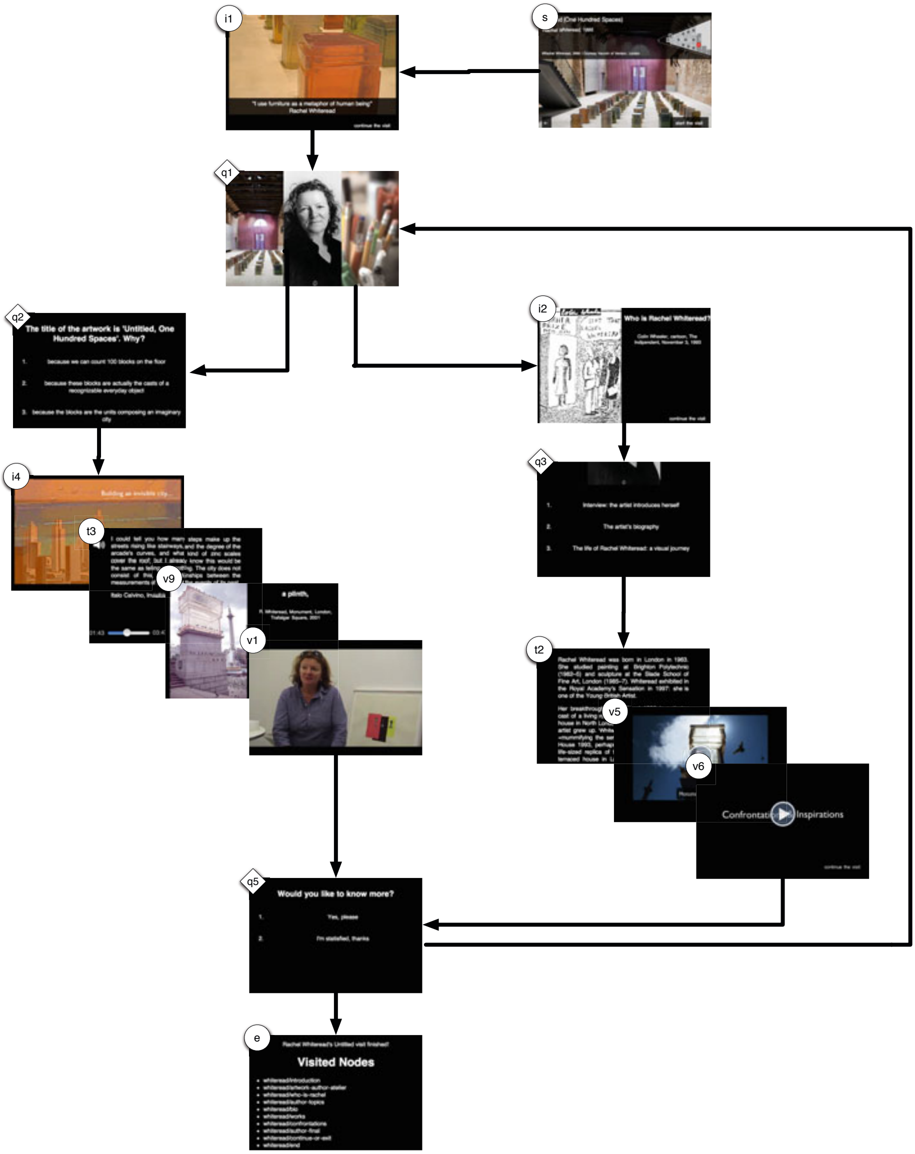
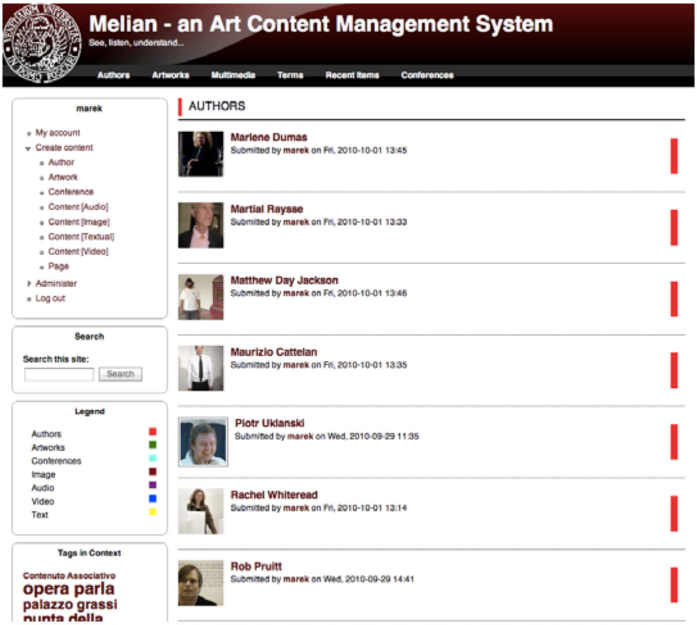

s
i1
"I use furniture as a metaphor of human being"
Rachel Whiteread
continue the visit
start the visit
q1
q2
The title of the artwork is 'Untitled, One Hundred Spaces'. Why?
1. because we can count 100 blocks on the floor
2. because these blocks are actually the casts of a recognizable everyday object
3. because the blocks are the units composing an imaginary city
i2
Who is Rachel Whiteread?
Colin Wheeler, cartoon, The Independent, November 3, 1993
continue the visit
i4
Building an invisible city...
t3
Italo Calvino, Invisible
v9
a plinth,
R. Whiteread, Monument, London, Trafalgar Square, 2001
v1
q3
1. Interview: the artist introduces herself
2. The artist's biography
3. The life of Rachel Whiteread: a visual journey
t2
Rachel Whiteread was born in London in 1963. She studied painting at Brighton Polytechnic (1982–5) and sculpture at the Slade School of Fine Art, London (1985–7). Whiteread exhibited in the Royal Academy's Sensation in 1997: she is one of the Young British Artist.
v5
v6
Confrontations Inspirations
continue the visit
q5
Would you like to know more?
1. Yes, please
2. I'm satisfied, thanks
e
Rachel Whiteread's Untitled visit finished!
Visited Nodes
whiteread/introduction
whiteread/artwork-author-atelier
whiteread/who-is-rachel
whiteread/author-topics
whiteread/bio
whiteread/works
whiteread/confrontations
whiteread/author-final
whiteread/continue-or-exit
whiteread/end

Fig. 4. A visit instantiated on the abstract topic graph of Figure 3



AUTHORS

Marlene Dumas
Submitted by marek on Fri, 2010-10-01 13:45

Martial Raysse
Submitted by marek on Fri, 2010-10-01 13:33

Matthew Day Jackson
Submitted by marek on Fri, 2010-10-01 13:46

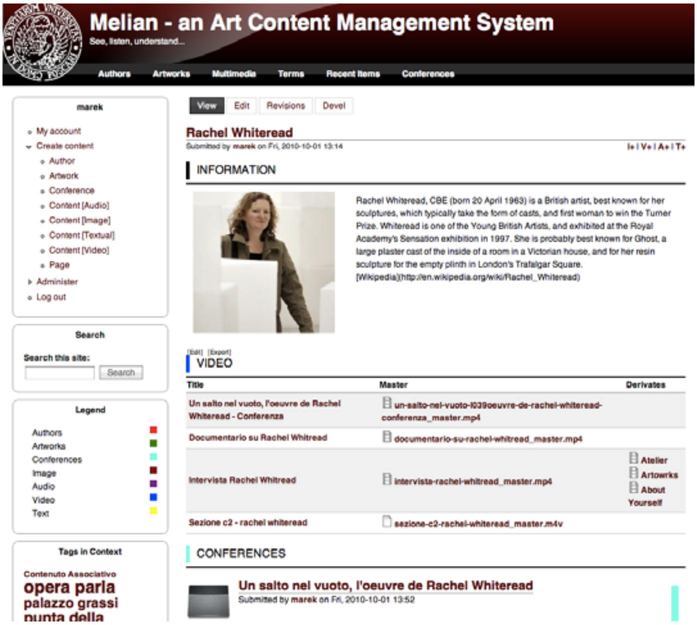
Maurizio Cattelan
Submitted by marek on Fri, 2010-10-01 13:35

Piotr Uklanski
Submitted by marek on Wed, 2010-09-29 11:35

Rachel Whiteread
Submitted by marek on Fri, 2010-10-01 13:14

Rob Pruitt
Submitted by marek on Wed, 2010-09-29 14:41

Fig. 5. The *CMS* artists page



Rachel Whiteread
Submitted by marek on Fri, 2010-10-01 13:14

INFORMATION

Rachel Whiteread, CBE (born 20 April 1963) is a British artist, best known for her sculptures, which typically take the form of casts, and first woman to win the Turner Prize. Whiteread is one of the Young British Artists, and exhibited at the Royal Academy's Sensation exhibition in 1997. She is probably best known for Ghost, a large plaster cast of the inside of a room in a Victorian house, and for her resin sculpture for the empty plinth in London's Trafalgar Square. [Wikipedia](http://en.wikipedia.org/wiki/Rachel_Whiteread)

VIDEO

Title	Master	Derivates
Un salto nel vuoto, l'oeuvre de Rachel Whiteread - Conferenza	un-salto-nel-vuoto-l039oeuvre-de-rachel-whiteread-conferenza_master.mp4	
Documentario su Rachel Whitread	documentario-su-rachel-whitread_master.mp4	
Intervista Rachel Whitread	intervista-rachel-whitread_master.mp4	Atelier Artowrks About Yourself
Sezione c2 - rachel whiteread	sezione-c2-rachel-whiteread_master.m4v	

CONFERENCES

Un salto nel vuoto, l'oeuvre de Rachel Whiteread
Submitted by marek on Fri, 2010-10-01 13:52

Fig. 6. The *CMS* page about Rachel Whiteread

in state t_2), and a visual journey of her early years as an artist. Two videos about her works (v_5) and a comparison with other artists (v_6) are finally presented.

To archive and classify data about the authors and artworks the domain experts used the *CMS* described in Section 3. A large quantity of contents have been inserted during the course of the project: our preliminary internal tests shows that the web application has been not only a way of storing the contents remotely, but also an instrument useful for sharing and collaboration. In Figure 5 a screenshot of the artists catalog (called *Authors* in the guide) of the *CMS* is shown. The page presents a view of all the nodes of type *Author* created during the project. In Figure 6 a screenshot of the page relative to Rachel Whiteread is presented. In the upper part the artist is introduced by a picture and a short biography. In the lower part a summary of the relationships with other content is shown: in the example the artist is in relation with four items of type *Video* and an item of type *Talk*. Note, in the lower part of the figure, the master/derivate organization of the multimedia files: the master video is the full interview, while the derivates focus on specific parts.

To create new nodes for the guide the domain-experts have to work with a data file that can be either prepared by hand, using a domain-specific language, or compiled with the aid of the *CMS*. In the data file the user specifies a set of *key-value* pairs. Each key identifies an element of the node template to be instanced with the given value. Different elements are instantiated in different ways: text is written in the file as a string, multimedia content is linked by specifying a URL, and so on. By writing a few simple statements the domain-experts can define the contents and questions nodes, specify their relationships, and select the multimedia files to include in the guide.

5 Conclusion

In this paper we have discussed our experience in designing a system for producing multimedia art guides for mobile devices as a joint effort of domain experts, interface and web designers and programmers; using the principles of End User Design, they can collaborate, each one according to own's expertise. We followed the *design during use* principle, setting up a system *adaptable* to end-users of different expertise, following the principle of *gentle slope*: modifications and personalizations can be carried out at different levels of increasing complexity involving domain experts at the simplest level and programmers only for substantial structural revisions.

We used the system to develop an interactive guide on the Apple iOS platform for the contemporary art collection of the François Pinault Foundation at Punta della Dogana in Venice. We have gained the evidence that articulating knowledge according to the user mood and need raises interest, curiosity and amusement, without diminishing the importance of art or trivializing its cultural value.

The web-based Content Management System we developed allows the project partners to archive, share, classify, and query multimedia contents enforcing a set of predefined guidelines: the process eliminate subjectivity and make the whole process less error-prone. The stored contents are accessible both through the web and a remote filesystem, resulting in an effective data representation for all the project participants, both of humanistic and scientific background.

The vast majority of art guides face the visitors with classical hierarchical menus to select contents: in this scenario the users have complete freedom to choose their tours. This approach, however, can be confusing to many, mainly in contemporary art that is, in general, considered an arduous topic. The *PuD* guide allows the visitors to create dynamic tours based on their interaction with the system. The user is solicited with a set of questions, in the form of text, images, gestures, etc., collectively defining the shape and the theme of the tour. In the proposed interaction model visitors are guided by the domain-experts choices toward a gradual learning and understanding process, according to their individual sensibility, curiosity, and feelings.

Embracing the end-user development principles we developed a three-level framework where domain experts, web designers and programmers can successfully and seamlessly cooperate to extend the guide features and contents. We plan to extend this concept by creating a more advanced *CMS*, where end-users can exploit a fully graphical interface to easily construct and visualize the topic graphs through touch or point and click events. The use of open standards ensures future portability on different devices. We would like that the *PuD* Project guide could be understood as a conceptual instrument that encourages the exploration of the art world from an inner view, particularly from the look of the artist creator.

Acknowledgements

The research project is a collaboration between the Department of Art History and Conservation of Artistic Heritage and the Department of Computer Science of Università Ca' Foscari Venezia with the François Pinault Foundation.

We acknowledge the contribution of Giuseppe Barbieri, director of the *PuD* guide project. Valentina Cefalù and Marco Del Monte, researchers in art critics and media, contributed to the conceptual design of the guide and to the multimedia material acquisition and organization.

A special thank to the persons who collaborated to the project: Martin Bethenaud, Marco Ferraris and Delphine Throuillard of Palazzo Grassi, Venice. Rob Pruitt and Gavin's Brown Enterprise, New York, Rachel Whiteread and Galerie Nelson-Freeman, Paris, Huang Yong Ping and Galerie Kamel Mennour, Paris, Jake and Dinos Chapman and Galleria Project B, Milan, Takashi Murakami and the François Pinault Foundation, Paris, Giandomenico Romanelli, Francesco dal Co and Eugenio Tranquilli.

The project is partially funded by Regione Veneto through the FSE program.

References

1. The MoMA iPhoneApp, `http://www.moma.org/explore/mobile/iphoneapp`
2. Ship Museum of Barcelona, `http://www.mmb.cat/`
3. San Jose Museum of Art (2005), `http://www.sjmusart.org/`
4. Gustav Klimt's works at Tate Liverpool Gallery (2008), `http://www.tate.org.uk/liverpool/exhibitions/gustavklimt/`
5. Barbieri, G., Celentano, A.: Multimedia technology: a companion to art visitors. In: Styliaras, G., Koukopoulos, D., Lazarinis, F. (eds.) Handbook of Research on Technologies and Cultural Heritage: Applications and Environments, pp. 393–410. IGI Global (2011)

6. Celentano, A., Orsini, R., Pittarello, F.: Towards an environment for designing and evaluating multimedia art guides. In: Proceedings of the International Conference on Advanced Visual Interfaces, pp. 93–96. ACM, New York (2010)
7. Costabile, M., Fogli, D., Mussio, P., Piccinno, A.: End-user development: The software shaping workshop approach. In: End User Development, pp. 183–205 (2006)
8. Damala, A., Kockelkorn, H.: A taxonomy for the evaluation of mobile museum guides. In: Proceedings of the 8th Conference on Human-Computer Interaction with Mobile Devices and Services, pp. 273–274. ACM, New York (2006)
9. Indianapolis Museum of Art. TAP tours: Content creation, standards and delivery tools for museum mobile tours, http://code.google.com/p/tap-tours/
10. Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-user development: An emerging paradigm. In: End User Development, pp. 1–8 (2006)
11. Maurizio, M., Cefalù, V., Monte, M.D., Celentano, A.: Building own's guided tour in a contemporary art guide. In: 3rd International Workshop on Personalized Access to Cultural Heritage PATCH 2011, in Conjunction with IUI 2011 Conference (2011)
12. Mayhew, D.J.: Principles and guidelines in software user interface design. Prentice-Hall, Inc., Upper Saddle River (1992)
13. Paternò, F.: Model-based design of interactive applications. Intelligence 11, 26–38 (2000)
14. Paternò, F., Campari, I., Scopigno, R.: The design and specification of a visual language: an example for customising geographic information systems functionalities. Computer Graphics Forum 13(4), 199–210 (1994)
15. Ronchi, A.: eCulture: cultural content in the digital age. Springer, Heidelberg (2009)
16. Stock, O., Zancanaro, M.: PEACH: intelligent interfaces for museum visits. Springer-Verlag New York Inc., New York (2007)

Beefing Up End User Development: Legal Protection and Regulatory Compliance

Patrick Kierkegaard[1,2]

[1] International Association of IT Lawyers, Hellerup, Denmark
[2] Eindhoven University of Technology, Eindhoven, The Netherlands
patrick.kierkegaard@iaitl.org

Abstract. The integral nature of IT to business processes means that every organization relies on technology to help manage the workflow, encourage innovation and maintain information flows. Agility has come in the form of end –user development, which allows users who do not have background in programming to develop or modify their own applications. Changes exist when users develop their own systems, creating a potential legal minefield. Many of the legal issues relate to copyright infringement and security breach. Aside from the potential liability for intellectual property infringement, end user development raises tort liability issues. Licensing plays a critical role as the tool used to protect rights and distribution condition. Industries have also chosen to restrict the damages which end users can recover through industry end –user licensing agreement, which exempts software publishers from all liability, which are often unacceptable to user. This paper investigates the legal issues surrounding end–user development, in particular copyright issues and whether liability for defects can be excluded through licensing agreement.

Keywords: liability, joint authorship, copyright infringement, data protection.

1 Introduction

The End User Development (EUD) landscape has undergone dramatic changes in the last ten tears. Traditionally, the major tool used for EUD has been spreadsheets followed by database management systems From being a simple technological tool enabling non-programmers to create software artefacts and then customize their models, recent developments such as the Web 2.0 technology, and social networking have allowed end user developers (EUDers) to create their own webpage, and wikis that enable collaborative creation, collective intelligence and distribution of content on the Internet and augmented user experience.

The EUD trend is now moving to Web 3.0, the next generation applications using natural language processing, machine based learning and intelligent applications. It combines the social elements of Web 2.0 with user-specific Semantic Web tools and opens possibilities for creating new tools. Web 3.0 allows users to roam freely from database to database, program to program. The idea is to make data to process and reuse.

The opportunities that will come after the web has been turned into a database are generating excitement to EUD. Integration of semantic web and social networks has

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 203–217, 2011.

also opened the possibilities of user generated networks, where EUDers can create their own networks by anyone at anytime and at any access points.

With the popularity of Web 2.0 in EUD and maturing of Web 3.0, their vulnerability to a host of legal issues also increases. And of course legal aspects are never simple! This paper will investigate the legal issues surrounding end – user development such as:

- Copyright and intellectual property rights infringement including trademarks
- Data protection, Retention and Security
- Choice of Law
- Jurisdiction
- Tort and Product Liability

2 End User Development

End – user development (EUD) is a *set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artefact* (Lieberman, 2006). Users who are inexperienced and unqualified as software developers are able to create their own solutions or to address a wide range of problems by adapting software systems on their own. Some of the tools used for EUD are: spreadsheets, scripts and macros, 3-D Models, Visual Programming, Web Pages, Animations and Game Modifications.

Initially, end users could only use the software without modification. Later, they were able to modify and adopt the systems on their own. The Open Source movement then revolutionized the software industry by opening up software systems for everybody's participation. This enabled non-professional developers to manage their local infrastructure of information systems. The advent of Web 2 .0 technology has also allowed EUDers to generate their own content and become a potential information provider. The use of social software (Facebook, blogs, wikis) has raised inevitable questions with regard to the legal issues based on three features of Web 2.0: ownership of creation and liability due to the interaction and collaboration; privacy and liability issue arising from loss or damage occurring.

The next generation of the world wide (Web 3.0) semantic web offers end-user developers the promise of a vast network of interconnected nodes of data, accessible to any computer and application connected to the internet. Semantic Web tools and techniques, for example, can be used to create a web based end-user programming environment. One of the main aims of Web3.0 is to make data available, accessed over the internet, and integrated with other data raising practical, legal issues to which their implementations are likely to give rise.

3 Legal Issue

3.1 Data Protection, Security and Confidentiality

Users generate posts, messages, chat, relationships with others, and in turn other people will add their own thoughts on this content. The Web 2.0 has brought a change in the way data is managed. This allows users to easily exchange or distribute information, images and file - often without requiring identification or validating information beyond a log-in and password.

New technologies have a dual effect. On one hand, these new tools enable more efficient working and innovation. On the other hand, they give rise to security problems. Vulnerabilities identified with this technology include: denial of service attacks, identity theft, worms, viruses, loss of privacy and malware. Inadequate privacy settings and security remain a major problem in end-user development. For instance, the number of incidents of malware distributed by social networking and emails are rapidly increasing. These attacks typically result in the compromised site directing malware at end-user computers and stealing sensitive data. Additionally, massive data loss and unintended release of data may result due to the EUDers negligence through file sharing of large sensitive files, which may find themselves in the public domain.

Collaborative file sharing often requires an analysis of data from multiple sources. Such analysis requires data integration, which enable integration with other environments. However, this requires appropriate tools which will maintain data integrity and are isolated from changes made by other members of the team. One of the major drawbacks of data integration is that other tools can access and potentially update the data and compromise data integrity.

Typically, end-user developers are not professional programmers and they are employed to perform other tasks. They write programs to assist their primary job. The problem is that they write software that may or may not work, but with little thought to "security." The goals of security are to ensure the confidentiality, integrity and availability of information while respecting fundamental human rights including privacy. Many EUDers write software with little knowledge of generally accepted good practices such as systematic testing. Warren Harrison (2004) wrote:

> *It's simply unfathomable that we could expect security... from the vast majority of software applications out there when they're written with little, if any, knowledge of generally accepted good practices such as specifying before coding, systematic testing, and so on.... How many X for Complete Idiots (where "X" is your favorite programming language) books are out there? I was initially amused by this trend, but recently I've become uneasy thinking about where these dabblers are applying their newfound knowledge.*

EUDers have a legal obligation to process all personal data is processed in a 'fair and lawful' manner and in accordance with the various data protection principles with the overriding aim of protecting the interests of the 'data subject'. Most countries have private and data protection laws. In the European Union, there exist several data protection and privacy legislations. The EU data protection legal framework mainly consists of the Data Protection Directive (95/46/EC), the e-Privacy Directive (2002/58/EC) and the Data Retention Directive (2006/24/EC), and several other Directives that set forth rules with respect to data protection.

Directive 95/46 is the reference text, at European level, on the protection of personal data. The Directive sets strict limits on the collection and use of personal data within the European Union and requires each Member States to have an independent national body responsible for protecting these data.

Any person acting under the authority of the controller or of the processor, including the **processor** himself, who has access to personal data, must not process them except on instructions from the controller. In addition, the controller must implement appropriate measures to protect personal data against accidental or unlawful destruction or accidental loss, alteration, unauthorised disclosure or access. All personal data must be:

- fairly and lawfully processed, -processed for specified purposes and not in any manner incompatible with those purposes,
- adequate, relevant and not excessive,
- accurate
- kept for no longer than is necessary,
- processed in line with the individual's legal rights,
- kept securely,
- transferred to countries outside the European Economic Area, only if the individual's rights can be assured.

In collaborative file sharing such as Wikis, the end user developer can act as both publisher and user of content. Other contributors can upload files, remove or edit contents that may contain personal information. Many Web 2.0 technologies also use tools which require users to register and submit personal information. Any obtaining, holding, processing and disclosing of personal information must be within the remit of the data protection and privacy principles.

In Case C-101/01 Criminal Proceedings against *Bodil Lindqvist,* the European Court of Justice held that the act of posting people's names, phone numbers, working conditions, and hobbies constitutes "the processing of personal data wholly or partly by automatic means," under article 3(1) of the DPD. Additionally, the court held, such processing of personal data is not covered by any exceptions in article 3(2) of the same Directive. Likewise, posting information in a website about a person's health, such as the parishioner's injured foot, constitutes personal data within the meaning of article 8(1) of the DPD. In the *Lindqvist* case, the defendant published the names and other information of a number of people working with her as volunteers for a parish of the Swedish Protestant Church. In her homepage, the lady also mentioned that one of the parishioners had an injured foot and was under medical leave. The Swedish lady did not inform the data protection supervisory authority nor notify the mentioned-parishioners about the existence of that webpage she created.

3.2 Intellectual Property Rights

End-user development poses interesting challenges relating to intellectual property rights. The term Intellectual Property (IP) covers a range of legal protections for creations of the human mind, such as trademarks, database, software, copyrights, patent, design rights and performers' rights.

Adaptation of works and collaborative creations raise profound issues for end-user development.

- Who owns the rights in works that are a result of collective collaboration?
- What are the risks associated with content reuse?
- Who is responsible for dealing with infringements within different legal jurisdictions and/or the identity of collaborators
- What may be permitted under exceptions to copyright

Wikis, for example, involve a number of authors who can claim copyright in their individual contribution. Texts are at the very core of copyright law, extending protection to original literary "works of authorship fixed in any tangible medium of expression."(CDPA 1988, ss 1-8; 17 U.S.C.A. §102(a) (1996) . Copyright forms the bedrock of the legal basis of what can, and cannot be done to text, images, sound recordings, film and broadcasts, and other types of content. Copyright gives the owner of copyright the right to authorise or prevent the doing of certain acts with a work protected by copyright. In particular, the owner can prevent third parties from copying, broadcasting and adapting the works. The right to adaptation is one of the exclusive rights of the copyright owner and therefore any adaptation should be with the permission of the copyright owner. Under the Berne Convention, protection covers all "literary and artistic works." This term encompasses diverse forms of creativity, such as writings, including scientific and technical texts and computer programs, databases that are original due to the selection or arrangement of their contents; musical works; audio-visual works (etc). Under article 10 of the TRIPS Agreement "Computer programs, whether in source or object code, shall be protected as literary works under the 1971 Berne Convention" (Kierkegaard, et al, 2010).

The categories of elements that will receive copyright protection are: text (data and code), digital images, computer generated works, and multimedia/database, and transformative works.

3.2.1 Collaboration: Joint Authorship

A question arises as to who owns the rights in works that are a result of collective collaboration in end user development. The essence of the collaboration agreement is copyright ownership. The formal legal definition of a "joint work" is "a work prepared by two or more authors with the *intention* that their contributions be merged into inseparable or interdependent parts of a unitary whole" (1976 US Copyright Act, Section 101).

Section 10(1) of the UK Copyright, Designs and Patents Act 1988 (and amending legislation) defines a work of joint authorship as "a work produced by the collaboration of two or more authors in which the contribution of each other is not distinct from that of the other author or authors." If it is not possible to distinguish exactly what each contributed, copyright will be owned jointly and no single contributor can publish or license the work without the consent of the other/two. If they are distinct, then the work is simply a collection or anthology of individual copyright works. When a joint work is created, the parties contributing to the work jointly hold copyright ownership unless the parties have agreed otherwise.

Many works have been inspired by another person's idea. However, merely a suggestion does not entitle a contributor to share in the copyright (*Robin Ray v Classic Plc* FM Plc [1998] FSR 622*)*. What copyright protects is not an idea, but the **expression** of it so that even the originator of the information that forms the basis of the work in question will not be considered the author of the work. There must be a contribution by *way* of *expression*, not mere facts or ideas. It is for this reason that copyright requires that the work must be fixed in a tangible form to qualify for protection, such as writing or recording.

In order to entitle someone to become a co-author, there must be a significant skill and labour in the **finished text** (actual written paper/work). In *Fylde Microsystems Ltd and Key Radio Systems* EWHC Patents 340 (11 Feb, 1998), the question arose as to whether copyright in software that had been written by the plaintiff with the defendant's co-operation belonged to the plaintiff alone or to the plaintiff and defendant jointly. It is not disputed that the plaintiff wrote the software. However, the defendant claimed joint authorship arguing that he invested the skill, time and effort into ensuring that the software performed in the way it now does through discussion of the software, saving the plaintiff a lot of time in perfecting the software.

In relation to the authorship, the Court addressed two issues:

- whether the defendant (putative author) has contributed the right kind of labour; and
- if he has, whether his contribution was big enough.

Although the Court found that there had been a close co-operation between the parties over 5 years and that the defendant had saved the defendant considerable time by testing the software - *that was not enough to claim authorship.*

"The defendant had put effort into error fixing and reporting faults and bugs. It had made a functional contribution by way of setting the specification for what the software was to do. It suggested causes of some of the faults in the software though it did not produce solutions to them. It had provided technical information concerning the hardware. It set parameters and timings within the software. ***Although, valuable and time consuming though such contributions must have been, they did not contribute to the authoring.*** *"*

The Court further elaborated*:*

"Beta testing does not make the user an author of a program and proof reading does not make the printer an author of a book. What counted was whether the defendant had contributed ***authorship skill.*** *"*

In the UK, collaboration between two or three people will result in a work of joint authorship only if their respective contributions to the finished work are not distinct from each other. However, if at the time of creation, the authors did not intend their work to be part of an inseparable whole, the result is a collective work. A "collective work" is a work, in which a number of contributions, constituting separate and independent works in themselves, are assembled into a collective whole. Collective works include compilations.

A compilation, which is formed by the collection and assembling of pre-existing materials or of data that are selected, coordinated, or arranged in such a way that the resulting work as a whole constitutes an original work of authorship. The copyright in each article remains distinct and separate from the copyright in and to the collective work as a whole. To create the collective work from the works of various authors requires the individual seeking to create the “collective work” to get permission from each copyright holder. In this case, **each author owns a copyright in only the material he or she added to the finished text.**

Protection for databases under copyright law is provided under the concept of a compilation copyright. Compilation copyrights protect the collection and assembling of data or other materials. In the case *of Feist Publications, Inc. v. Rural Telephone Service Company, Inc.,* the U.S. Supreme Court ruled that a compilation work such as a database must contain a minimum level of creativity in order to be protectable under the Copyright Act. Under 17 U.S.C. § 101 of the US Copyright Act, a compilation is defined as a "collection and assembling of pre-existing materials or of data that are selected in such a way that the resulting work as a whole constitutes an original work of authorship." In the European Union, directive 96/9/EC on the legal protection of a database defines database as “a collection of independent components, such as pieces of information, data, or works, arranged in a systematic or methodical way and which are individually accessible by electronic or other means. “The content of a database can include literary, musical, artistic, or other works or material such as text, sound, images, numbers, facts, and raw data.

Multimedia resources also fall under this definition but computer programs do not. Wiki-like directories using user-generated contents is regarded as a database .Ward Cunningham, the developer of WikiWikiWeb, the first Wiki software, described a wiki as “*the simplest online database that could possibly work*” and as such, enjoy database protection. There are two main bases for database protection: copyright protection (under the condition of creativity) and new sui generis protection (under the condition of substantial investment).

Another issue arises concerning the ownership of the innovative tools if the work is created in the course of the EUDers ‘employment. If the work is made by an employee in the course of his employment, his employer is the first owner of any copyright in the work *subject to any agreement to the contrary* (see Sec. 11(1) of UK Copyright, Designs and Patents Act). The same rule applies in the US and many other jurisdictions. According to the copyright law in the United States , a work created by an employee within the scope of his or her employment is a work made for hire. The employer for whom the work is made is the "author" of the work for copyright purposes and is the owner of the work's copyright (unless the employee and employer have agreed otherwise).

Many domain experts use their free time to develop or modify creative tools during their free time. The question arises as to the ownership of the technology or data. The work made for hire rule does not give employers ownership of works made by employees outside the scope of their employment. This means that the end user developer who wrote the text or created the new technology on his own initiative on weekends or outside office hours owns the copyright. The work for hire rule does not apply as this was done outside the scope of his employment. If the company wants ownership, it needs to acquire an assignment *i.e.,* a transfer of copyright ownership. An assignment is not valid unless it is in writing and is signed by the rightholder.

However the rules governing employment relationship are not always straightforward. There are situations where the end user developer writes a program in his free time using the code which was developed at work. If the programming is done on work-supplied equipment or under instructions, the work *may* fall within the course of employment. In many instances, the domain expert might work remotely outside working hours to create a work for his employer. Others might transfer to other companies bringing with them the tools they developed in their previous employer. The employee can make a claim of ownership to the imported materials are used or embedded in software written for the new employer.

Determining who is the copyright owner could be problematic in the absence of a written agreement or provisions in the terms of employment that clearly state that the "course of employment" includes work undertaken for the employer whether that work is conducted on or off the employer's premises and whether during or outside normal working hours.

If the work is created by an independent contractor and the independent contractor signs a written agreement stating that the work shall be "made for hire," the commissioning person or organization owns the copyright. Commonly, however, the contract is silent on the matter. Under the Copyright, Designs and Patents Act 1988, there **is no automatic vesting of copyright in a commissioned work in the commissioner of such a work.** If there is no written agreement specifically addressing copyright ownership between a producer and an independent contractor, the independent contractor is generally presumed to retain copyright ownership in those elements of the work that the contractor created. In *Robin Ray v Classic FM* [1998] *FSR 622,* the UK Court of Appeals held that a contractor providing services owns the intellectual property in the materials created for the client in the absence of an implied or express term. Justice Lightman's view is that in order to imply some grant of rights, it is necessary to have either a licence or an assignment of the copyright.

Further complications arise when the independent contractor contracts the work to third parties. A written agreement between the commissioner and the contractor may not afford total protection.

3.2.2 Prerequisite for Copyright

The *sine qua non* of copyright is that in order for the domain expert to claim copyright, he has to satisfy the element of originality and copyrightability of his data and text. In the US, 17 U.S.C. § 102(a) of the Copyright Act defines the scope of copyright protection: "Copyright protection subsists…in *original* works of authorship fixed in any *tangible medium of expression*." Thus a work must be original to the author in order to be protected. The term "original" as used in copyright law simply means (i) that the work was independently created by the author (as opposed to copied from other works), and (ii) that it possesses at least some minimal degree of creativity(*Feist Publications, Inc. v. Rural Tel. Serv. Co.*, 499 U.S. 340, 361 (1991).

Establishing originality implicates only a light burden. As noted by the US Fourth Circuit in *Universal Furniture v. Collezione Europa* (Aug. 20, 2010):

> *." "[T]he requisite level of creativity is extremely low; even a slight amount will suffice. The vast majority of works make the grade quite easily, as they possess some creative spark."*

Even when the work at issue is a compilation of pre-existing design elements, the originality threshold remains low: *"Copyright protection may extend to such a compilation, even if the material of which it is composed is not copyrightable itself. . . ; it is sufficient if original skill and labour is expended in creating the work."*

Names, titles, and short phrases or expressions are not subject to copyright protection. Even if a name, title, or short phrase is novel, distinctive, or lends itself to a play on words, it cannot be protected by copyright. To be protected by copyright, a work must contain at least a minimum amount of authorship in the form of original expression. Names, titles, and other short phrases are simply too minimal to meet these requirement. However, there are also many exemptions to this rule. If the phrase is a whole or heart of the work, or that that literary phrase must be so idiosyncratic that its appearance in another work would preclude coincidence, then it could be protected. In *Brilliant v. W.B. Productions, Inc*. Civ. No. 79-1893-WMB (S.D. Cal Oct. 22, 1979), a company copied two of Brilliant's phrases -"I may not be totally perfect, but parts of me are excellent" and "I have abandoned my search for truth and am now looking for a good fantasy"-and altered a third phrase, all for sale on t-shirt transfers. The court acknowledged that the phrases were distinguished by conciseness, cleverness, and a pointed observation, and ruled that they were protected by copyright. In short, the author must either demonstrate that the phrases exhibit sufficient creativity.

The second prerequisite for copyright protection is that that works must be fixed in a tangible form of expression. The fixation requirement requires a physical embodiment, of a work which is in its essence intellectual, that is, intangible. Tangible form may include anything written on paper, saved to disk (web pages, graphics on web, electronic mail messages or computer programs), or saved on any audio/video device. There are, however, several fundamental items that are not eligible for copyright protection: ideas, facts, titles, names, procedures, and works not fixed in tangible form. Copyright only protects the form in which these ideas or facts are expressed, not the ideas or facts themselves.

3.2.3 Digital and Visual Images

Multimedia works include any combination of music, text, diagrams, graphics, illustrations, photographs and audiovisual imagery combined into an integrated presentation, along with accompanying projection and playback equipment. In the UK, the copyright law protects images and photographs which are considered to be artistic works. It also protects films, broadcasts and sound recordings. In general, one will need permission to copy such works for educational purposes unless the works whose copyright protections have expired enter into the public domain and thus can be used without permission.

Like other copyright works, it may be possible to copy images under one of the fair dealing exceptions without infringing copyright provided. For example, an EUDer could copy an image which is made available in the public domain for the purposes of giving criticism, parody and review of the image provided a suitable acknowledgment is given.

3.2.4 Computer-Generated Works

Depending on the jurisdiction, approaches to ownership of computer-generated works vary. Under the UK Copyright, Designs and Patents Act, 1988, "In the case of a literary, dramatic, musical or artistic work which is computer-generated, the author shall be taken to be the person by whom the arrangements necessary for the creation of the work are undertaken." This would imply that "a user of a word-processor or excel spreadsheet would own exclusive rights in his or her respective essays or worksheet compilations. A user of a word-processor would be solely responsible for the "arrangements necessary for the creation of the work, and thus, would ostensibly acquire copyright protection in the resulting output.” (Glasser, 2001) The problem arises when there is little human input in the creation of a computer-generated program.

In *Express Newspapers plc v Liverpool Daily Post & Echo plc* [1985], the High Court ruled that ‘output from a computer that has been randomly generated by the machine itself is a copyright work’. The case arose out of the battles between the national newspapers in their lottery competition. In this case, a computer program was used to generate unique five letter sequences which were printed on 22 million cards as part of a competition called Millionaire of the Month. Council for the defence argues that as there was no human author, copyright did not subsist – hence the defendant was free to publish the winning sequence is their newspaper. Whitford J Ruling defined the role of the computer as instrumental, saying:”The computer was no more than a tool” and rejected the defence argument stating ‘it would be to suggest that, if you write your work with a pen, it is the pen which is the author of the work rather than the person who drives the pen.” In the ruling the author of the work was adjudged to be the programmer.

However, in *Nova Productions Ltd v Mazooma Games Ltd* [2006], the owner of a coin-op pool game sued two competitors who created games with similar features. The games looked and played differently and none of the source code for the original game was copied. The court concluded that there had been no copyright infringement. Although certain features of the defendants’ games were to some extent derived from or inspired by Pocket Money, there was not reproduction of a substantial part of any of the classes of copyright work relied on. “It confirms the approach taken in *Navitaire* to infringement of copyright in source code. It will clearly continue to be difficult, if not impossible, to extract high level functional or behavioural architecture from source code and match it to what is said to have been taken by the alleged infringer”(Smith et al, 2006).

The decision also makes it difficult for software copyright owners to take action against developers of software which has the same functionality, but which does not copy the underlying code or the graphics displayed on screen.

3.2.5 Transformative Works

A transformative work takes a previously established work of art and turns it into something new. An EUDer can use the wiki engine, graphics, or other elements to create a new expression which is no longer a solely an entry. The creation of such transformative work that is truly a **fair use** is not an infringement of copyright, and therefore does not run afoul of US Copyright Law even if a protection mechanism is circumvented. The doctrine of fair use (17 U.S.C. § 107) allows limited use of copyrighted material without requiring permission from the rights holders. In fact, the fair use exceptions have been credited for more than for more than $4,500 billion dollars in annual revenue for the United States economy representing one-sixth of the total U.S. GDP. (Computer and Communications Industry Association, 2008) In determining whether the particular use is fair use, four factors must be weighed:

- the purpose and character of your use;
- the nature of the copyrighted work

- the amount and substantiality of the portion taken. However, even if you take a small portion of a work, the copying will not be a fair use if the portion taken is the "heart" of the work; and
- the effect of the use upon the potential market.

For a work to be considered transformative instead of derivative, it must demonstrate originality under the law. A derivative work simply takes the work and changes a few aspects of it. In other words, the works must be obviously substantially different for the change to be considered transformative under law.

In *Kelly v. Arriba Soft Corporation and Perfect 10, Inc. V Amazon,* the US Ninth Circuit found that the use of thumbnail images was allowed under the copyright law doctrine of "fair use." The courts find their use (for indexing purposes) as transformative because it provides an added benefit to the public.

3.2.6 Derivative Works

According to the Section 101 of the U.S. Copyright Act of 1976:

> *"A 'derivative work' is a work based upon one or more pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations, elaborations, or other modifications, which, as a whole, represent an original work of authorship, is a 'derivative work'".*

In short, all other modifications whose outcome is a new, creates a new copyright on all original aspects of the new version. In *Campbell v. Acuff-Rose Music, Inc,* the court held that a commercial parody can qualify as fair use and qualifies as derivative work. The lightning rod was 2 Live Crew allegedly parodic use of the "Pretty Woman" song. The court reasoned that "even if 2 Live Crew's copying of the original's first line of lyrics and characteristic opening bass riff may be said to go to the original's "heart, "that heart is what most readily conjures up the song for parody, and it is the heart at which parody takes aim." The Supreme Court concluded that 2 Live Crew produced otherwise distinctive music.

If an EUDer takes the copyrighted source code of any program and actually revises, modifies, changes or translates it into another computer language or new program, he has created a derivative work of the program. The touchstone for a derivative work is the "recasting, transforming, or adapting" of the original work, often to a new form, the copyright extends only to the material contributed by the author and not to the pre-existing material. The real measure of the derivative work is that it be recognizable. (See *Sega Enterprises Ltd. v. Accolade, Inc.*, 977 F.2d 1510 (9th Cir. 1992).

Open source software allows derivative works (developers can reuse existing code to fill specific needs rather than write new software from scratch). The GNU General Public License is one of the best known and most widely used licenses governing open source code. It is a means of implementing a concept called **Copyleft**.

Copyleft attempts to negate copyright for the purposes of collaborative software development. When an author releases a piece of code under the GPL license, she is granting recipients of the source code the following rights:

- anyone can use the code anywhere, in any situation;
- anyone can redistribute the code to anyone else, as long as the source code is included and the distribution license remains the GPL;
- anyone can create a derivative work of the code and redistribute it, as long as the resulting source code is also made available at redistribution time, and as long as the resulting source code is licensed under the terms of the GPL.

The GPL implies that a derivative work is one that is linked, statically or dynamically, with the original work. The GPLv2 refers several times to derivative work, however, it fails to provide a definition of what constitutes derivative work and thus raises questions. The word derivative does not appear in the GPLv3 and is replaced by the expression ("work based on").It allows the user to convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code provided that he meets all of these conditions, specifically the provision of Notices.

3.2.7 Trademarks

A trade mark is a sign which serves to distinguish the goods and services of one organisation from those of another. A trade mark may consist of any sign capable of being represented graphically, particularly words, including personal names, designs, letters, numerals, the shape of goods or of their packaging, provided that such signs are capable of distinguishing the goods or services of one undertaking from those of other undertakings. In order to serve as a trademark, a mark must be distinctive -- that is, it must be capable of identifying the source of a particular good.

Wikis abound with logos and EUDers can potentially infringe trademark law by posting trademarks that are subject to protection. Trademark infringement claims generally involve the issues of likelihood of confusion, counterfeit marks, and dilution of marks. To be more specific, the use of a trademark in connection with the sale of a good constitutes infringement if it is likely to cause consumer confusion as to the source of those goods or as to the sponsorship or approval of such goods.

Dilution can occur if the character of the trademark becomes clouded by an unwanted association, either through tarnishment, or through blurring, which means the use of a famous trademark causes consumers to blur the two companies in their minds. Dilution by blurring is considered to have occurred when a trademark is used by someone other than the trademark owner on products that are very different from those normally produced by the trademark owner (for example Coke yoyo), as the key element of harm to distinctiveness is present resulting in the weakening of the strength of their marks.

4 Licenses

Copyright gives the owner of copyright the right to authorise or prevent the doing of certain acts with a work protected by copyright. One of the primary values of owning copyright rights is the ability to transfer some or all of those rights to third parties. A transfer of copyright rights is usually either an assignment or a license. A license on the other hand, encapsulates the rights and privileges that a copyright holder grants to someone else *vis a vis* something they have created, such as authorization to reproduce, prepare derivative works, distribute, perform or display, in the manner specified in the license. In a collaborative environment such as wikis and open source software, where many EUDers are collaborating to build something, a single piece of open source software will have multiple copyright holders. To get around these difficulties, people and organisations often enter into negotiations for licences (contracts) to give them permission. In an open source software, the organisation such as the Free Software Foundation requires every contributor to assign the copyright of their creation back to them through a license agreement. These licenses are governed by contract law. Some of the licenses available are:

- General Public License version 3.0 (GPLv3) grants the rights to copy, modify and distribute the program.
- GNU General Public License (GPL) allows everyone to make changes or extend the source code and redistribute it as long as the changes are clearly marked, and the modified work is also licensed.
- The BSD license allows for a redistribution and use of source code and object code with or without modification. It also allows the licensee to mix closed source software with open source software without the GNU GPL's limits on integration.
- The Creative Commons License system (consisting of six licenses) which offers a set of standardised and automated licences those authors can affix to their work in order to indicate under which conditions the work may be used.

However, the problem is that that many of the licenses are onerous and contain unreasonable terms. The licenses contain a liability disclaimer and a warranty exclusion clause. They do not contain the kinds of representations and warranties of quality or fitness for a particular purpose that commercial software vendors sometimes negotiate into agreements among themselves. No warranty of any kind is offered to the licensee, and the software is provided "*as is*". Most licenses provide that no damages arising from the use of the software, including any direct or indirect damages (e.g. loss of use, data or profits, business interruption), will entail the liability of the Licensor. The outright denial of any form of warranty protection for users shifts to the user any and all liabilities for product failure as well as intellectual property infringement. (Kierkegaard, 2010).

In the EU, these disclaimers are actually ineffective against statutory liabilities, when they are not in conformity with consumer protection or product liability. While in the USA, many manufacturers often include "limit of liability" clauses in contracts with buyers, the producer cannot do so in Europe regardless of what contractual arrangement they have made with the injured person. Directive 85/374/EEC imposes a scheme of strict product liability for damage arising from defective products. However, this liability only applies to damages caused to the physical integrity of the person and to his/her property. If someone in Europe is injured by the device made by the EUDers or if the EUDer is injured by the software, the injured person is required to prove the damage, the defect in the product and the causal relationship between the two, and has three years to seek compensation.

Moreover, when a consumer is party to an open source license, the contract provision limiting the liability of the licensor will be in violation of the Unfair Contract

Terms Directive ((1993/13/EEC) since the standard tern causes a significant imbalance in the parties' rights and obligations arising under the contract, to the detriment of the consumer. Terms excluding or limiting the legal liability of a seller or supplier in the event of the death of a consumer or personal injury to the latter resulting from an act or omission of that seller or supplier is unfair.

The European Union Public License (EUPL) is the first European Free/Open Source software (F/OSS) license with emphasis placed on the universality of the license.. The license was developed to conform to the copyright law of the 27 member states of the European Union unlike most existing licenses originating from the United States, which are based only on American law. To be valid in all Member States, limitations of liability or warranty had to be precise, and not formulated "to the extent allowed by the law" as in most licenses designed with the legal environment of the United States in mind.

To strengthen consumer protection, the EU Commission has proposed a new law that would hold developers responsible for the security and efficacy of their product by extending the principles of consumer protection rules to cover licensing agreements of products like proprietary and open-source software or other licensed content. *But how does a customer prove that it was the EUDers' application, for example, and not another application causing the problem*?

4.1 Choice of Law and Jurisdiction

Because of the ease with which materials can be copied and distributed, EUDers face the difficulties of determining the lex contractus and lex fori,. This is mainly due to the fact that Internet encompasses numerous jurisdictions and the copyright and contract laws of many countries differ significantly.

Most licenses contain terms which specify the forum and law applicable in case of disputes such as copyright infringement, violation of licenses and tort. However, EUDers based in different countries could be collaborating without any contractual agreement. Any contributor could put infringing materials or share software which causes damage or harm. All of the contributors will be liable for damage. Moreover, in case of litigation, most of the considered licenses are governed by US law and therefore prejudice the public policy provisions that are enjoyed by the consumer in his country of residence. In addition, many licenses determine the jurisdiction competent for any litigation, and are normally the courts in California or Paris.

In conflict of laws, the choice of law rules for tort are intended to select the lex causae by which to determine the nature and scope of the judicial remedy to claim damages for loss or damage suffered. In the European Union, the applicable law and jurisdiction are governed by the Rome II Regulation and Brussels 1 Regulation.

The Rome II Regulation creates a harmonised set of rules within the European Union to govern choice of law in civil and commercial matters (subject to certain exclusions) concerning non-contractual obligations, including specific rules for tort/delict, unjust enrichment , *negotiorum gestio* and *culpa in contrahendo.*

Of significance to EUDers would be *the* law applicable to a non-contractual obligation arising out of a tort/delict. It shall be the law of the country in which the damage occurs. Where the product causes damage, the law applicable to a non-contractual obligation arising out of damage caused by a product will be *lex domicile*, or the law of the country in which the person sustaining the damage had his or her habitual residence when the damage occurred.

In non-contractual obligations arising from an infringement of an intellectual property rights, the law of the country for which protection is claimed applies (Art.8).

Council Regulation 44/2001(Brussels I) lays down the basic rules governing the jurisdiction of courts in civil and commercial matters. The basic principle is that jurisdiction is to be exercised by the EU country in which the defendant is domiciled. Regardless of domicile, in case of disputes arising from the registration or validity of patents, trademarks, designs or other similar rights, the courts of the EU country in which the deposit or registration has been applied for, has taken place or is under the terms of a Community instrument or an international convention deemed to have taken place is the law of forum. In matters relating to liability for wrongful acts - tort, delict or quasi-delict, these will be decided by the courts for the place where the harmful event occurred or may occur.

5 Conclusion

Changes exist when users develop their own systems, creating a potential legal minefield. Many of the legal issues relate to copyright infringement and security breach. Aside from the potential liability for intellectual property infringement, end user development raises tort liability issues. Licensing plays a critical role as the tool used to protect rights and distribution condition. Industries have chosen to restrict the damages which end users can recover through industry end –user licensing agreement, which exempts software publishers from all liability, which are often unacceptable to user and in contravention of the EU legal frameworks on product liability, consumer protection in unfair contract terms. In order to avoid legal pitfalls, the process of considering IPR and data protection should ideally be built in to every stage of creating or repurposing content found on the Web.

References

Computer and Communications Industry Association. Fair Use Economy Represents One-Sixth of U.S. GDP (September 12, 2007), http://Ccianet.org, Retrieved from http://web.archive.org

Kierkegaard, P., Adrian, A.: Wikitopia: Balancing Intellectual Property Rights within Open Source Research Databases. Journal of Computer Law & Security Review 26(5) (2010)

Glasser, D.: Copyrights in Computer-Generated Works: Whom, If Anyone, Do We Reward? Duke Law and Technology Review 0024 (2001)

Harisson, W.: The Dangers of End User Programming. IEEE Software (July-August 2004), http://eusesconsortium.org (Retrieved 12-20-10)

Lieberman, H., Paternò, F., Wulf, V.: End User Development. Kluwer Publishers, Dordrecht (2006)

Smith, G., Chan, N.: Computer Copyright Claims Fail. Bird and Bird (2006), http://www.twobirds.com/ (Retrieved December 20, 2010)

Part III
Short Papers

Light-Weight Composition of Personal Documents from Distributed Information

Danilo Avola, Paolo Bottoni, and Riccardo Genzone

Department of Computer Science, "Sapienza" University of Rome, Italy
{avola,bottoni}@di.uniroma1.it, riccardo.genzone@gmail.com

Abstract. Digital information is typically distributed across several resources, but users are annoyed by the need to deal with different formats and applications for its retrieval, processing and creation. Corporate solutions for document composition are heavy-weight and unwieldy for extemporaneous usage. On the contrary, we propose a light-weight interaction framework, in which document specification results from the selection of existing resources, also comprising annotations retrieved from the Web. Style-sheets allow on-the-fly generation of actual documents.

Keywords: Document composition, Web annotation.

1 Introduction

When preparing reports, curricula, technical papers or even personal memos, users retrieve and access public or private information contained in resources of different nature (e.g. formatted documents, post-it notes, complex web pages) and distributed across different locations (e.g. owned folders, corporate databases, or the Web). In many cases, the information relevant to a user's task is only a part of a rich document, possibly composed of heterogeneous contents. Moreover, the different kinds of resource containing the target information force users to use several complex specialized applications to both extract relevant parts from the documents and compose them in a suitable way.

This situation poses two types of problems. First, users want to retrieve information, without being interested in its location or format, or in the tools needed to access it. Second, users need to collate pieces of information from multiple, possibly heterogeneous, sources, and to generate documents on-the-fly from such collections, without committing to any specific application, but retaining information on the composition of the document. While corporate solutions exist, they are typically heavy-weight applications, so that users need to learn a different tool or adhere to some specific discipline for managing information.

We address end-user needs for organising information by proposing a light-weight resource-oriented interaction framework for document composition. The framework is based on the *Universal Resource Engine* (URE), allowing users to: 1) create complex resources by combining data of heterogeneous nature and different provenience; 2) organise information as resources stored in personal

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 221–226, 2011.

pools; 3) publish resources, making them available to other users' pools in a collaborative setting; 4) retrieve resources from pools; 5) generate documents on-the-fly, using stylesheets to produce different renderings or formats for the same content. In particular, in this paper we show how resources derived from Web annotations can be integrated within URE.

Paper organisation. Section 2 reports work on document composition and modeling of multimedia resources. Section 3 introduces a scenario of document composition related to personal interests, while Section 4 illustrates URE architecture and the interactive activities it supports. Section 5 concludes the paper.

2 Related Work

The issue of relieving users from the need to use several tools for document composition has been addressed, at the industrial level, by companies such as StreamServe, IsisPapyrus, or PrintSoft[1]. Their solutions are based on proprietary templates and allow a complete management of the whole document composition life-cycle. Solutions based on XML for content storage, description and manipulation, e.g. Thunderhead[2], are gaining momentum.

Zotero[3] is a Web-based framework for exploring resources and identifying related content according to their contextual environment (e.g. the web page), thus presenting aspects common to both URE and the MADCOW (for Multimedia Annotation of Digital Content Over the Web) annotation system adopted in this paper. Being completely Web-based, Zotero is mostly oriented to the construction of reference libraries, rather than the organization of personal information.

Wiki-based technologies allow rapid integration of content, but not a direct trace to its origin. RDF is being adopted for document composition to manage content through tagged descriptions, often adopting reasoning mechanisms based on Description Logic [1,2], Document Composition Logic [3], or other logics [4]. While URE's modular structure can be adapted to use different logics for document retrieval, we consider here recursive composition of multisets of complex resources as the basis for document construction and exploration.

We base our modelisation of resources on the OMMMA (Object-oriented Modeling of MultimMedia Applications) metamodel [5], mapping the types of resources in URE to OMMMA types. However, OMMMA does not directly define the concept of annotation, which describes a subtype of textual resource in URE. Interesting relations can be drawn with the metamodel for content repurposing proposed in [6], allowing the description of several aspects related to features of specific media and of interaction processes.

[1] http://www.streamserve.com,http://www.isis-papyrus.com, http://www.printsoft.com

[2] http://www.thunderhead.com

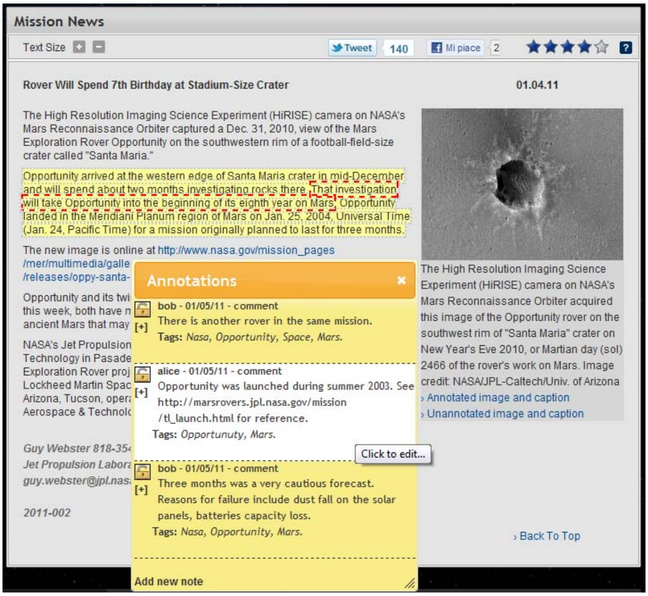
[3] http://www.zotero.org/

3 A Scenario

The scenario illustrates how the resource-based features of URE can be employed to integrate publicly available annotations on Web pages (produced with the MADCOW system [7]) in the construction of complex documents.

John is a student with a passion for everything is space exploration and is a member of a local club of space enthusiasts, for which every month he prepares a small report on current activities. To this end, he exploits URE to manage a pool of resources on space exploration. He maintains a list of sources, that he constantly updates, and from which he extracts several pieces of information, which he makes available to URE by saving them as resource items that he categorises under a number of topics. As a frequent visitor of websites on these topics, he is also a user of MADCOW, that he employs to generate annotations on the visited pages and to explore public annotations by other users.

To prepare for the meeting, John collects updates from preferred websites. While browsing the NASA page on the Mars Rover mission, he notices annotations made by two MADCOW users (see Figure 1), providing context information and decides to incorporate this material. To find out other interesting annotations, he launches a query from URE to the MADCOW server for annotations on the page. The server replies with the text and the metadata associated with annotations matching the query. He incorporates these as resources in the resource pool and starts previewing them and selecting the most interesting ones. Finally, he writes a new summary piece, creates a resource to keep comments, and incorporates these in a complex resource to create and print a memo.



Rover Will Spend 7th Birthday at Stadium-Size Crater

01.04.11

The High Resolution Imaging Science Experiment (HiRISE) camera on NASA's Mars Reconnaissance Orbiter captured a Dec. 31, 2010, view of the Mars Exploration Rover Opportunity on the southwestern rim of a football-field-size crater called "Santa Maria."

Opportunity arrived at the western edge of Santa Maria crater in mid-December and will spend about two months investigating rocks there. That investigation will take Opportunity into the beginning of its eighth year on Mars. Opportunity landed in the Meridiani Planum region of Mars on Jan. 25, 2004, Universal Time (Jan. 24, Pacific Time) for a mission originally planned to last for three months.

The new image is online at http://www.nasa.gov/mission_pages
/mer/multimedia/galle
/releases/oppy-santa-

Opportunity and its twi
this week, both have n
ancient Mars that may

NASA's Jet Propulsion
Technology in Pasade
Exploration Rover proj
Lockheed Martin Spac
Arizona, Tucson, opera
Aerospace & Technolo

Guy Webster 818-354
Jet Propulsion Labora
guy.webster@jpl.nasa

2011-002

bob - 01/05/11 - comment
There is another rover in the same mission.
Tags: *Nasa, Opportunity, Space, Mars.*

alice - 01/05/11 - comment
Opportunity was launched during summer 2003. See http://marsrovers.jpl.nasa.gov/mission/tl_launch.html for reference.
Tags: *Opportunuty, Mars.*

bob - 01/05/11 - comment
Three months was a very cautious forecast. Reasons for failure include dust fall on the solar panels, batteries capacity loss.
Tags: *Nasa, Opportunity, Mars.*

The High Resolution Imaging Science Experiment (HiRISE) camera on NASA's Mars Reconnaissance Orbiter acquired this image of the Opportunity rover on the southwest rim of "Santa Maria" crater on New Year's Eve 2010, or Martian day (sol) 2466 of the rover's work on Mars. Image credit: NASA/JPL-Caltech/Univ. of Arizona
› Annotated image and caption
› Unannotated image and caption

› Back To Top

Fig. 1. MADCOW annotations on a NASA page for the Opportunity mission

4 Managing Resources

We call *universal resource* any entity uniquely identifiable and accessible within a computer-based system and whose availability is limited in time or space. Typically, they are abstract digital counterparts of resources (or portions thereof) actually existing in the physical world, or virtual entities accessible through the system. Besides being identified, resources are described through metadata, each of which offers a criterion for retrieval purposes. Several universal resources can refer to the same physical resource and in principle an arbitrary number of universal resources can be generated having as origin the same physical resource.

The URE environment adopts a two-level architecture where the OMMMA metamodel provides the types for user-side activities of resource selection, composition and manipulation, while a resource-oriented programming language (in the current implementation the WIPPOG rule-based language [8]) allows the computational specification of these activities. With reference to Figure 2, the *Composer* system consists of: 1) an Aggregation Support (AS) environment with primitives for checking resource availability and managing generation, editing and deletion processes; 2) a Universal Resource Engine (URE) able to resolve references to simple and complex resources, imported or generated in a resource pool; 3) a Composition Support (CS) with tools to convert resource contents into a format readable by a suitable generator of documents; 4) a collection (URPs) of pools of currently available Universal Resources.

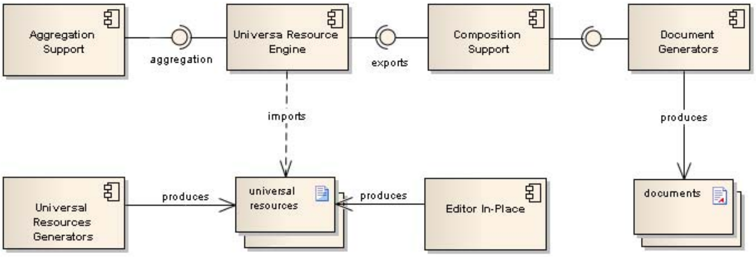


Fig. 2. Overall architecture for URE

URE architecture is complemented by three types of components: 1) **Document Generators**, able to obtain a resource description from a URP and produce a physical document; 2) **In-place Editors**, interactive light-weight tools for producing simple textual or graphical documents; 3) **Universal Resources Generators**, external applications able to export resources in the URE format, or software services able to explore repositories of resources and import them into a URP. Through the URE GUI, users specify the resources they are interested in, explore the structure of the documents they are composing, and activate dedicated viewers. URE has been customized to interact with the MADCOW public interface supporting read-only queries to retrieve information about public annotations. The query field contents are sent, in XML format, to MADCOW,

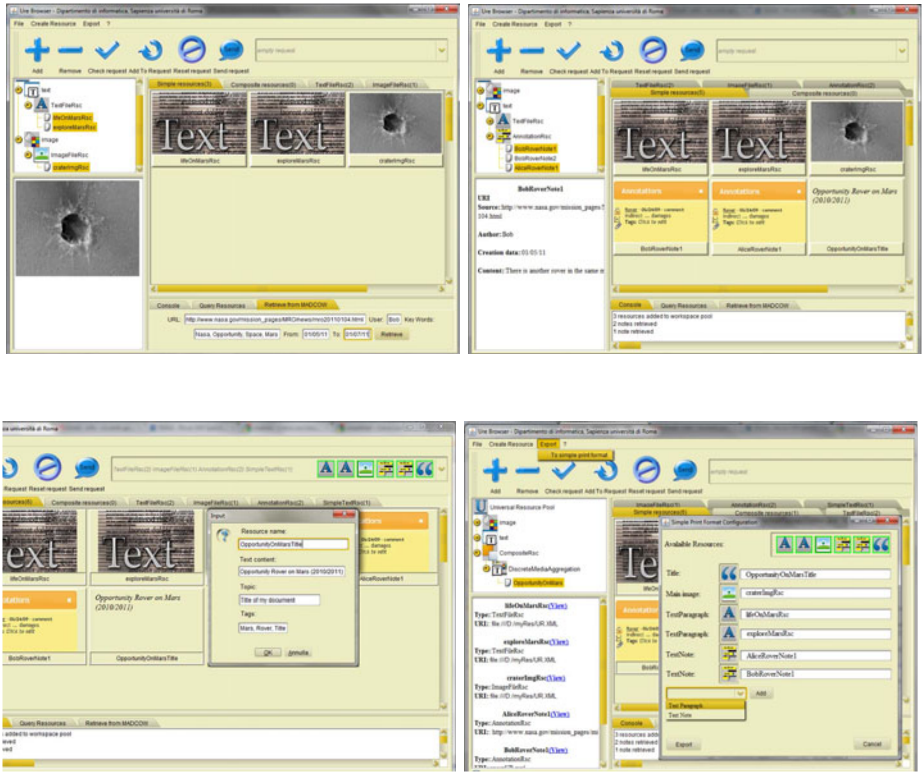


Fig. 3. Retrieving, incorporating, and composing resources

which parses them to build a suitable query for the MADCOW's database. The result of the query is packaged in a new XML file and sent to URE.

Considering the OMMMA metamodel, simple resources are directly related to `Media` subtypes, while a URE-based interactive composition process defines realisations of `ApplicationUnit` by aggregating resources into complex ones.

Within URE, resources can be retrieved from a pool according to metadata descriptors, or tags associated with them. They are presented to the user, who can decide to upload them to the working pool. Figure 3 (top left) shows the results of a query for retrieving simple resources with material on the *Opportunity* mission, within the resources (both text and image) already present in the resource pool. In the bottom panel of the interface, a query to MADCOW is prepared to find all the annotations available for the NASA page devoted to the Opportunity mission. The retrieved annotations are encapsulated as annotation resources and downloaded to the resource pool for inspection. The user can then create a complex resource from selected ones. In the bottom left subfigure of Figure 3, John has examined the available resources and selected six of them (an image and two text files, together with two annotations and a text fragment), shown as thumbnails, to create the document. He has also saved the resulting composite resource and specified its metatags for future queries. Finally, Figure 3 (bottom right) shows a summary of the composite structure named OpportunityOnMars, which John can export for print. In this case, John is exploiting the simple print format configuration, provided as default in URE.

5 Conclusions

URE is a system for browsing, organising and retrieving multimedia resources and composing them into complex structures, from which documents can be generated on-the-fly. URE can access resources distributed over different pools in a collaborative setting, and is able to query the MADCOW database to obtain annotations over Web pages of interest. In general, it aims at integrating material which was produced at any moment for whatever reason, but which can relate to the current interests of a user. This allows end-users to compose complex documents without recurring to heavy-weight applications incorporating facilities for creation, editing and integration of richly formatted documents. Current developments are aimed at integrating in URE a number of light-weight editors for images and text, for creation of simple resources as need arises, and at achieving a full integration with MADCOW, so that resources can be used to provide content for annotations.

References

1. Fensel, D., van Harmelen, F., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F.: OIL: Ontology infrastructure for semantic web. IEEE Int. Syst. 16(2), 38–45 (2001)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation, Applications. CUP, Cambridge (2003)
3. Chen, S.-K., Wu, K.-L., Yih, J.-S.: Decoupled common annotations for reusing XML document composition logic. In: Proc. IASTED CSEA, pp. 678–683 (2004)
4. Almendros-Jiménez, J.M.: An RDF query language based on logic programming. ENTCS 200(3), 67–85 (2008)
5. Sauer, S., Engels, G.: UML-based behavior specification of interactive multimedia applications. In: Proc. HCC 2001, pp. 248–255. IEEE Computer Society, Los Alamitos (2001)
6. Obrenovic, Z., Starcevic, D., Selic, B.: A model-driven approach to content repurposing. IEEE Multimedia 11(1), 62–71 (2004)
7. Bottoni, P., Civica, R., Levialdi, S., Orso, L., Panizzi, E., Trinchese, R.: MADCOW: a multimedia digital annotation system. In: Proc. AVI 2004, pp. 55–62 (2004)
8. Bottoni, P., De Marsico, M., Di Tommaso, P., Levialdi, S., Ventriglia, D.: Definition of visual processes in a language for expressing transitions. JVLC 15(3), 211–242 (2004)

Really Simple Mash-Ups

Yvonne Dittrich, Peter Madsen, and Rune Rasmussen

IT-University of Copenhagen
Rued Langaards Vej 7
2300 Copenhagen S, Denmark

Abstract. Mash-ups are applications – typically web applications – designed by combining data from several web services into a new tool or expression. New mash-ups emerge every day. Different End-User Development environments for mash-ups are available. However, the identification of mashable web-services, their exploration and the definition of data aggregation from heterogeneous web services still requires familiarity with programming representations and environments. Can the provision and usage of mashable web-services become as easy as the provision and usage of RSS feeds? What is needed is (1) a web server that allows providers of mashable web services to define data-formats describing their interface and users of the same web services to identify mashable data sources and (2) a simple standard to annotate the result of web services so that mash-ups editors can support even non professional developers in pruning and aggregating the data according to their needs. The article presents our bid on how this vision can be realized.

Keywords: Mash-ups, Cooperative End-User Development, Web-services.

1 Introduction

Mash-ups composed from existing web services into new functionality, are becoming a more and more appreciated way of personalizing home pages, creating picture galleries of your own photos and providing different ways of presenting e.g. search results. Several end-user environments, like Yahoo Pipes, allow users to easily combine known building blocks.

However, the discovery, exploration and aggregation of new mashable web services is still a bottleneck. This is especially relevant for mash-ups combining heterogeneous data sources, e.g. in data source, consumer, or enterprise mashups [8]. And maybe this is the reason why only few services are used in mash-ups. Existing standards – like the Dublin Core Metadata – that could support the data aggregation are cumbersome both for suppliers and users of mashable data sources. Is there a possibility to make the provision usage of mashable web services as easy as the provision and presentation of an RSS-feed?

This short paper presents the result of our exploration of this question. It does not address the End-User interface of mashing-up services but focuses on an infrastructure that supports the sharing and combination of mashable services through Mashup editors like Marmite [9]. The following section explores the existing possibilities for

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 227–232, 2011.

sharing mashable web services and support for aggregation of results. Thereafter we present our concept, consisting of (1) a web server supporting a community process defining data formats for provision and consumption of web services, the publication of mashable web services, and the possibility for developers to search for suitable web services to use for their mash-up ideas; and (2) a standard for annotation of web services so that the results of different web services can easily be aggregated. Our proposal of a RSM standard and infrastructure is supported by two prototypes; the RSM web server allowing for definition of data formats and a query by example based editor demonstrating how the information in the RSM annotations to web services can be used for aggregating data.

2 Existing Support for Sharing of Mashables and Data Aggregation

Mash-ups are not a new phenomenon, and data exchange through web services and other techniques has been done and supported for a long time. Below, we shortly discuss the three initiatives closest to our vision.

programmableweb.com
The programmableweb.com [1] is a web site allowing to upload and download both mashable services and mash-ups. The site lists at the time of writing 2151 mashable services, ‘APIs’. Each API is described with a short comment. The statistics indicate that only a fraction of the uploaded mashables is used by the mash-ups uploaded on the same site. The activity on the web site shows that it provides a well-received location service for APIs and for mash-ups. However, there is no support for potential users of mashable services to explore the interface prior to accessing the service.

The Microformat Community
The microformats community [7] defines and shares xhtml-based formats for marking web content in order for automatically being able to identify and collect information e.g. about events from webpages and blogs. Part of the idea presented here is doing the same for mashable web services. On top of the sharing of formats, functionality to locate mashable services and a common way to describe the results of web-services to allow for data aggregation is needed.

The Resource Description Framework and the Dublin Core Metadata
The Resource Description Framework (RDF) is a metadata format developed by the World Wide Web Consortium (W3C). [2] The RDF provides an ontology and knowledge management inspired notation allowing to describe the contents of web documents in the form of ‘subject-predicate-object’ statements. A net of such statements can be depicted as a tree.

The Dublin Core Metadata Initiative develops and maintains an ISO standard for data exchange that is formulated using the RDF notation. [3] On first glance, it sounds like a good idea to use these standards: RDF schemas and descriptions can be formulated in XML. There exist a XML version of the Dublin Core Metadata (DCM). Why not just annotate the result of web service calls with the suitable RDF description and use the description to merge and aggregate data from different web services? There already exists a standard for 'Semantically Annotated' web services: SA-REST. [4] The output from web-services is translated, 'lifted', to a common RDF schema, merged, pruned, aggregated, and translated back, 'lowered', to serve as input for another service. A more lightweight version, WSMO-Lite, is presented in [10]: web services can be annotated with among other information interface, functional and behavioral specifications.

Discussion

The RDF schemes have the reputation of being unnecessarily verbose. [5] The logic of subject-predicate-object triples might provide an additional hinderance for providing a for programmers of mashable services easily accessible representation for data merging, pruning and aggregation.

For the exchange of business critical data, like e.g. electronic invoicing or the like, this approach is a necessary one. However, for a developer who contemplates to make his recent fun development available for others, the effort to understand the logic of RDF schemes and find a fitting ontology would provide an overhead he might not want to take. Even if the developer does not need define and maintain 'lifting' and 'lowering' schemes this might provide a hinder to annotate his service.

An additional problem is that the DCM cannot easily be extended. The DCM has the status of an ISO standard. That means, there is a committee that is steering the development of the standard. New schemes undergo review and revisions before they are finally included in the standard and published for usage. If no fitting RDF scheme is available, one would need to submit an addition to the right standardization committee. Furthermore, the usage of RDF, SA-REST, and maybe the DCM standards does not help with sharing and locating mashable services.

Is there a possibility to combine the advantages the three above presented approaches? Can we have a lightweight definition of formats combined with a *really* lightweight annotation mechanism and a location service for mashable services?

3 Really Simple Mash-Ups

The discussion above indicates that the existing solutions are not satisfying for a simple way to share mash-ups and support the aggregation of data from heterogeneous services.

What would be needed is the possibility to (a) define common formats in a community process without too much overhead and organizational red tape, (b) easily publish mashable web services; (c) to match the existing interfaces of web services with the common format in case that the interface cannot be adjusted; (d) locate services and information about the format they use; and (e) access to the meta-data necessary to aggregate the output.

The Really Simple Mash-up concept consists of a web server, the global RSM dictionary, solving (a), (b) and (d), and a format for annotations of web services solving (c) and (e). Both the RSM dictionary and the annotation format depend on the concept of an RSM-key, a way to specify the format of an atomic piece of data. All three ingredients are presented below. Feedback is welcome.

3.1 RSM Keys

RSM keys are defined in the global RSM dictionary and are used to annotate the result of web services. RSM keys are defined slightly different in the annotated documents than in the dictionary:

Attribute	Used in RSM dictionary	Used in annotated document
Name	Required	Required
Tag	Required	Required
Format	Required	Optional
Descriptor	Required	Not applicable
Value	Not applicable	Required
Local Name	Not applicable	Optional
Id	Required	Required

Name. The name should refer to the concept the data describes, e.g. “GPSposition”. The RSM Dictionary guarantees the uniqueness of the name.

Tag. Tags are a non-rigid way of grouping and relating RSM tags. Its tags will relate a RSM key to one or more real world domains.

Format. This element contains the regular expression used to validate whether a specific value complies with the format of a RSM key. In document annotations it supports validation of values without access to the RSM dictionary.

Descriptor. Descriptors can be used to assign descriptive information to a RSM key, like date of creation, author, date of modification, or description. A key can contain an arbitrary amount of Descriptor elements.

Value. This element contains the actual key value. It is used to annotate a specific data set.

Local Name. A local name can be assigned for an annotation to describe a local appliance of a key concept, e.g. a RSM key with the Name “GPSposition” could have a local name of “InterestPoint” either to distinguish it from another usage of “GPSposition” in the same document or simply to help mash-up creators.

Id. Unique identification of a RSM key, required in both annotation and definition.

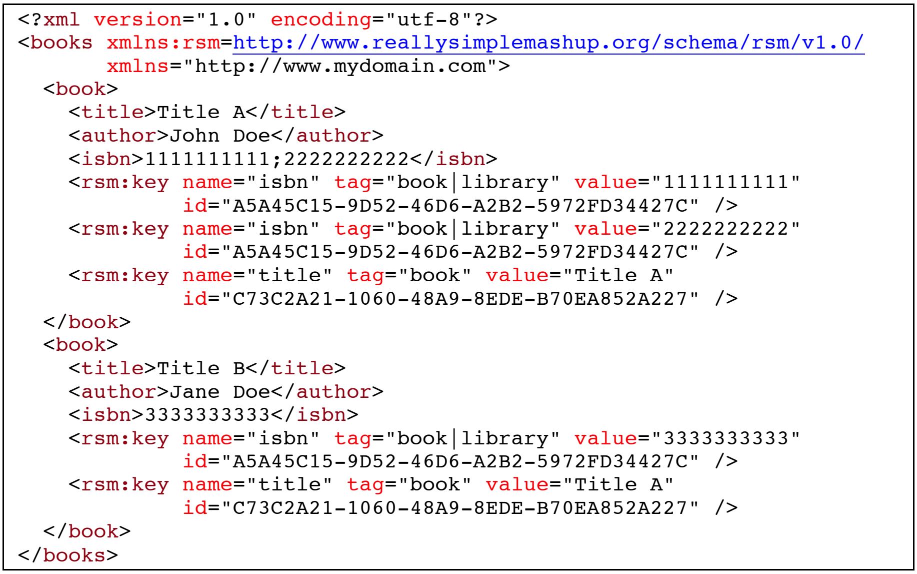
3.2 A Global RSM Dictionary

The global RSM dictionary needs to support three basic needs or use cases: The definition of RSM keys and their formats, the location of keys for annotation of mashable web service results, the sharing and the location of mashable web services.Tags denoting application domains are used to group and relate RSM keys in a non-rigid way. Rather than applying a screening or review process it supports reaching a core set of commonly used RSM keys by inviting users to reuse concepts already defined. The only restriction is that the RSM key names should be unique.

The RSM dictionary will encourage data providers to register their services and link them to the relevant domains as well as the RSM-keys they use. It might expose information about the usage to help data providers choose the most accepted RSM-keys when annotating their data services. The RSM dictionary should hold additional searchable information to assist the process of identifying RSM enabled mashable web services. For a more thorough presentation of the design see [6]. A prototype for the RSM dictionary is available from the authors.

3.3 Annotating Web Service Results

We propose to use an annotation approach rather than ask providers to change the interface of their services. The web services might have been designed based on in-house policies or to fit with specific local requirements. The overhead for providing the services to other users should be kept as low as possible. We therefore follow the SA-REST and WSMO-lite strategy to enrich the services with RSM keys.



```
<?xml version="1.0" encoding="utf-8"?>
<books xmlns:rsm=http://www.reallysimplemashup.org/schema/rsm/v1.0/
       xmlns="http://www.mydomain.com">
  <book>
    <title>Title A</title>
    <author>John Doe</author>
    <isbn>1111111111;2222222222</isbn>
    <rsm:key name="isbn" tag="book|library" value="1111111111"
             id="A5A45C15-9D52-46D6-A2B2-5972FD34427C" />
    <rsm:key name="isbn" tag="book|library" value="2222222222"
             id="A5A45C15-9D52-46D6-A2B2-5972FD34427C" />
    <rsm:key name="title" tag="book" value="Title A"
             id="C73C2A21-1060-48A9-8EDE-B70EA852A227" />
  </book>
  <book>
    <title>Title B</title>
    <author>Jane Doe</author>
    <isbn>3333333333</isbn>
    <rsm:key name="isbn" tag="book|library" value="3333333333"
             id="A5A45C15-9D52-46D6-A2B2-5972FD34427C" />
    <rsm:key name="title" tag="book" value="Title A"
             id="C73C2A21-1060-48A9-8EDE-B70EA852A227" />
  </book>
</books>
```

Example of a XML document after RSM annotation.

Web service results can come in different notations. By defining the RSM keys notation independent of the format, we open up for defining annotations for different notations. Above we present our annotation format for XML. A similar one can be designed for JSON or any other API format.

4 Discussion and Conclusion

The Really Simple Mash-Up infrastructure proposed here provides a low overhead and low bureaucracy approach to sharing formats for mashable web services. It supports acquisition and aggregation of data from heterogeneous web services when designing new mash-ups. The definition of RSM keys and formats is left to the users of the RSM dictionary in a true web 2.0 spirit. By proposing an annotation of the web service results rather than asking providers of mashable services to change the interface, we minimise the additional effort for providers of services.

As the focus was on data acquisition and aggregation, we did not address how to use the RSM keys for specifying input parameters. Another extension would be to provide the functionality of the RSM dictionary as a mashable service, described by RSM keys. Mash-Up editors could then seamlessly include the RSM functionality.

Any design, however, has to be tested with real users and real applications. We did only develop a very basic proof of concept prototype (e.g. there is presently no support for registration and location of RSM-annotated services). If the RSM dictionary and annotation format would be adopted and used by a broader community the design would for sure be refined. The concrete design of the web site, the keys, and the annotation format might be up to discussion. The basic ideas, however, we believe will hold beyond our proof of concept prototypes.

Acknowledgements

Thanks to the anonymous reviewers for their valuable feedback.

References

1. The Programmable Web, `http://www.programmableweb.com/` (last accessed January 2011)
2. Ressource Description Framework (RDF), W3C Semantic Web Activity, `http://www.w3.org/RDF/` (last accessed January 2011)
3. Dublin Core MetaData Initiative, `http://www.dublincore.org/` (last visited January 2011)
4. Sheth, A.P., Gomadam, K., Lathem, J.: SA-REST: Semantically Interoperable and Easier-to-Use Services and Mashups. IEEE Internet Computing 11(6), 91–94 (2007)
5. Horrocks, I., Schneider, P.P., van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. J. of Web Semantics 1, 7–26 (2003)
6. Madsen, P., Rasmussen, R.: Really Simple Mash-Ups. Final project on the Diploma in IT program. Supervised by Y. Dittrich. IT University of Copenhagen (2009)
7. Microformats, `http://microformats.org/` (last accessed March 2011)
8. Wong, J., Hong, J.I.: Making mashups with marmite: towards end-user programming for the web. In: Proc. of CHI 2007, pp. 1435–1444. ACM, New York (2007)
9. Lee, C.-J., Tang, S.-M., Tsai, C.-C., Chen, Y.-C.: Toward a new paradigm: Mashup patterns in web 2.0. WSEAS Trans. Info. Sci. and App. 6(10), 1675–1686 (2009)
10. Vitvar, T., Kopecky, J., Zaremba, M., Fensel, D.: WSMO-Lite: Lieghtweight Semantic Descriptions for Services on the Web. In: Proc. of the 5th ECOWS, pp. 77–86. IEEE, Los Alamitos (2007)

Teachers as Designers: Enabling Teachers to Specify Dynamic Web Programming Projects for Students

Mary Beth Rosson, Elizabeth Thiry, Dejin Zhao, and John M. Carroll

Center for Human-Computer Interaction
The Pennsylvania State University
University Park, Pennsylvania 16803 USA
mrosson@psu.edu

Abstract. wConnect is a participatory action research project [1] building a developmental community for women in computer and information science (CIS). One activity hosted by wConnect is end-user web programming workshops for high school girls. Thus far we have relied on our lab's Java-based toolkit for these workshops, but this requires considerable advance work and subsequent maintenance by team members for each workshop, making the outreach process difficult to generalize and sustain. Currently we are using the Drupal CMS to create a web authoring tool that will support these workshops but also enable teachers to design and specify their own project concepts. Once a project is specified, learners can instantiate and complete them, adding data and other customizations according to personal preferences. We describe our work thus far on the authoring tool, design tradeoffs we are facing, and ongoing research.

Keywords: End-User Web Development, K-12 Education, Content Management Systems, Sustainable Action Research.

1 Introduction

As girls enter their teenage years, many lose interest in computer and information science (CIS); in the U.S. the number of women graduating with CIS degrees has dropped by almost 25% in the past ten years [2]. This trend threatens the future availability of qualified CIS professionals, and particularly the diversity and vitality of the profession. The *wConnect* project has been addressing this problem through professional community building. Many young women hold a narrow and negative view of CIS professionals as "geeks" who work alone on boring computer programming tasks [3]. To counter such misconceptions, wConnect is building social networks that convey a broader and more personally meaningful view of CIS.

Earlier publications have described the wConnect online community and its outreach activities, including web programming workshops [4][5][6]. In this paper we report new work motivated by *sustainability* goals. In particular we are exploring authoring tools that would enable teachers or other interested parties to design and conduct outreach workshops of the sort demonstrated by wConnect members. In the balance of this short paper we motivate these new efforts, describe how the tool works, and conclude with a brief discussion of open issues and ongoing work.

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 233–238, 2011.

1.1 The wConnect Developmental Community

wConnect is a *developmental community* – members join with shared commitments to personal growth and helping one another transition through stages in this growth [7]. The specific learning domain is education and careers in CIS, and the growth includes changes in both cognitive skills (e.g., analyzing and building software) and attitudes (e.g., view of individuals and activities that comprise the CIS profession). The stages comprise middle or high school girls who are tend to be disinclined toward CIS; 1^{st} – 2^{nd} year undergraduates who are considering CIS education or careers; 3^{rd} – 4^{th} year undergraduates who have committed to a CIS degree; and female alumni or mentors who are in the CIS workforce.

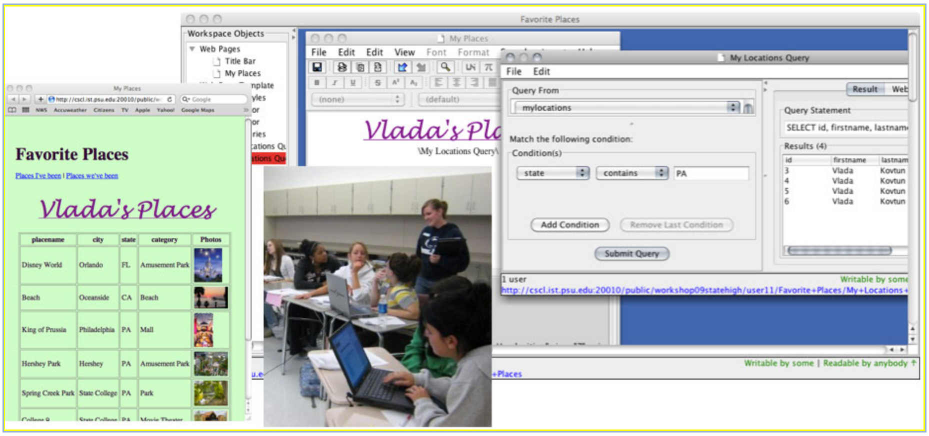
In the past three years, wConnect has developed as a participatory action research project [1]. We began with a core team of undergraduate members who first created content for access-controlled community websites and activities and gradually became more directly involved in the development of the online system. We worked with three different infrastructures in this – the Bridgetools research toolkit; the Facebook API; and currently the Drupal CMS ([8] articulates this evolutionary process).

The development and use of the online community has been an important focus for wConnect. It currently supports a variety of interactions, including user profiles with developmental status, blogs, discussion forums, online chat and instant messaging, group and subgroup email sending, a video library (e.g., interviews with alumni), photo and image sharing, a bi-monthly newsletter, games and playful visualizations.

1.2 High School Workshops as an Outreach Activity

In addition to online activities, wConnect members organize events in the physical world, primarily outreach workshops for high school girls. These workshops typically take about 90 minutes and require preparatory work by an undergraduate volunteer, who contacts the school, identifies a teacher advocate to help with local details, works out logistics, and leads a team of peers to conduct the actual workshop. wConnect has sponsored seven such workshops outside the university and four within. The outcomes have been promising, with both school staff and students enthusiastic about the outreach event, the girls showing basic comprehension of the concepts taught, and about half of them (about 50 thus far) expressing interest in staying connected [9].

As a developmental activity, the workshops have two pedagogical goals. On the one hand, the development and delivery of the activity is a growth experience for current wConnect members; on the other the activities are designed to convey dynamic web development concepts and skills to the high school girls: they learn to add data records to a database; construct queries that return subsets of the data; and build web pages that display not only static content like text and images, but also dynamic content delivered through embedded database queries. Thus far, these goals have been supported by the Bridgetools workspace seen in Fig. 1; in this environment, the database, each query, and each web page are instantiated as independent objects in a Java-based interactive environment. These objects can be edited directly in the workspace (on the right of the figure), but can also be rendered as objects on the web (see the web page on the left). Working with these objects, the girls grasp the essential concept of a dynamic web page that draws information from a server-based data store.



Favorite Places
Workspace Objects
Web Pages
Title Bar
My Places
My Places
http://cscl.ist.psu.edu:20010/public/w
NWS Accuweather Citizens TV Apple Yahoo! Google Maps
File Edit Edit View Font Format
My Locations Query
File Edit
Result
Web
Writable by some | Readable by anybody
+Places

Favorite Places

Places I've been | Places we've been

Vlada's Places

placename	city	state	category	Photos
Disney World	Orlando	FL	Amusement Park	
Beach	Oceanside	CA	Beach	
King of Prussia	Philadelphia	PA	Mall	
Hershey Park	Hershey	PA	Amusement Park	
Spring Creek Park	State College	PA	Park	
College 9	State College	PA	Movie Theater	

Query From: mylocations

Match the following condition:

Condition(s): state contains PA

Add Condition | Remove Last Condition | Submit Query

Query Statement: SELECT id, firstname, lastnam

Results (4)

id	firstname	lastnam
3	Vlada	Kovtun
4	Vlada	Kovtun
5	Vlada	Kovtun
6	Vlada	Kovtun

1 user

http://cscl.ist.psu.edu:20010/public/workshop09statehigh/user11/Favorite+Places/My+Locations+

Writable by some

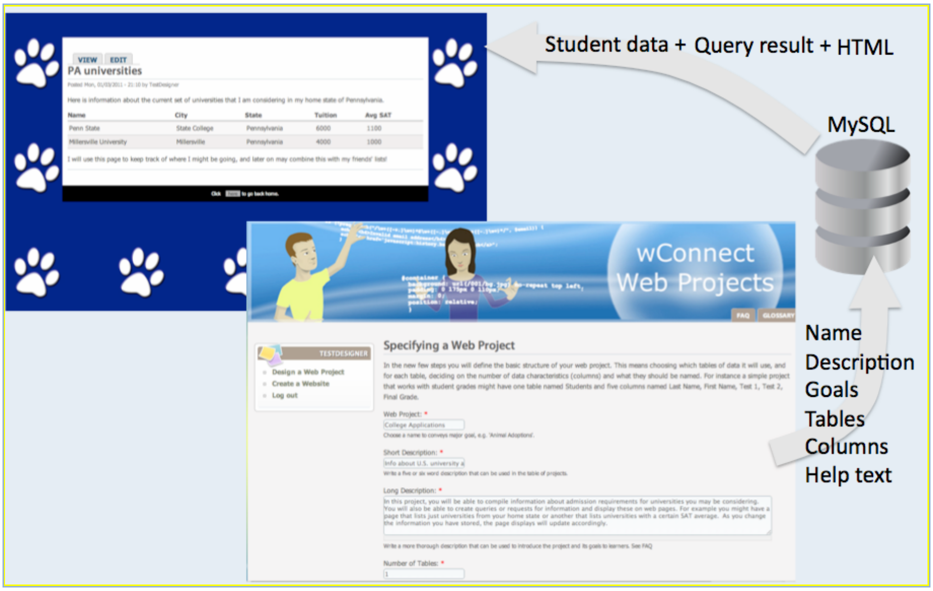
Fig. 1. High school web programming workshop as supported by Bridgetools. The workspace used to edit content objects is right and left, the lower shots show a website and class setting.

The Bridgetools infrastructure has allowed us to convey the basic concepts of dynamic web programming, but has raised concerns about longterm sustainability of this activity within wConnect. First, the Java toolkit is complex and wConnect community members do not have the skills to debug or extend it when problems arise. Second, it is difficult or impossible to access the lab server through the firewalls intalled by some schools. Third, the initialization of a Bridgetools workshop is complex and tedious, with separate instantiation of database tables, user accounts, and project objects for each prospective user. Finally, Bridgetools expertise is limited to our research group, making it impossible to generalize and share the activity in other education contexts. For these reasons, we have begun to explore an alternative workshop infrastructure using Drupal, and open-source content management system (CMS).

2 wProjects: A Drupal-Based Authoring Tool for Web Projects

We chose to explore Drupal (drupal.org) as the foundation for a new tool because the current wConnect community website was built with this technology and as a result, our membership has a growing body of relevant expertise. We also wanted leverage the Drupal community, both for maintenance of the infrastructure and the influx of novel functionality. We confirmed that Drupal could support the basic learning goals (populate a table with data, query the data, and embed the results in a web page) and began protoyping a new tool that we dubbed wProjects (for web projects).

As we refined wProjects, we recognized a bonus of the Drupal CMS: we can support both specification and use of projects *with the same web client*. We did this by defining two possible roles for users, project designer and project learner. Designer-users can access project definition dialogs; learner-users can only instantiate and edit projects. As a result we have expanded our target population of end users to include project designers. Other work discusses the goals and experiences of learners [5][6]; in this short paper we focus on project specification by teachers or other mentors.



PA universities

Posted Mon, 01/03/2011 - 21:10 by TestDesigner

Here is information about the current set of universities that I am considering in my home state of Pennsylvania.

Name	City	State	Tuition	Avg SAT
Penn State	State College	Pennsylvania	6000	1100
Millersville University	Millersville	Pennsylvania	4000	1000

I will use this page to keep track of where I might be going, and later on may combine this with my friends' lists!

Specifying a Web Project

In the new few steps you will define the basic structure of your web project. This means choosing which tables of data it will use, and for each table, deciding on the number of data characteristics (columns) and what they should be named. For instance a simple project that works with student grades might have one table named Students and five columns named Last Name, First Name, Test 1, Test 2, Final Grade.

Web Project: *
College Applications
Choose a name to conveys major goal, e.g. 'Animal Adoptions'.

Short Description: *
Info about U.S. university a
Write a five or six word description that can be used in the table of projects.

Long Description: *
In this project, you will be able to compile information about admission requirements for universities you may be considering. You will also be able to create queries or requests for information and display these on web pages. For example you might have a page that lists just universities from your home state or another that lists universities with a certain SAT average. As you change the information you have stored, the page displays will update accordingly.
Write a more thorough description that can be used to introduce the project and its goals to learners. See FAQ

Number of Tables: *
1

Fig. 2. Basic architecture of the teacher's web project specification process in wProjects. Key project parameters (tables and their structure; learner prompts) are collected and stored in a database, then later used to instantiate and guide construction of project instances.

2.1 Specifying Projects Using wProjects: The Basic Concept

An overview of the project specification process is in Fig. 2. At the heart of wProjects is Drupal's MySQL database that stores the specifications for projects (e.g., number and structure of the data tables) and the data corresponding to learner-user project instances (e.g., table data added by a student, queries saved). As soon as a project is specified, it becomes available in template form for learners to select and use.

The user experience is quite different in the Drupal tool than in Bridgetools. Rather than operating in a workspace that "contains" a variety of interactive and editable objects, all work is done in a web browser via a tabbed user interface that provides access to a number of special-purpose forms. For instance, the initial page in the project specification dialog appears in the middle of Fig. 2, where the teacher is giving the project a name, a shorthand description, a longer description that expresses the hir or her learning goals, and the number of tables it will use.

2.2 A Simple Scenario

We do not have space to provide a complete wProjects walkthrough with illustrations. Instead, we present a step-by-step scenario, again focusing on the teacher-designer role during project specification. In the scenario, imagine that Ms. Kissell is a teacher designing a "College Applications" project; her students will use the project to gather and post information about universities they are considering.

1. Ms. Kissell specifies the name of the new project, along with a shorthand description, a longer description that summarizes its goals, and the number of tables the project will use (just one table for this project).
   This information is stored in variables; the names and number of tables are used to generate custom versions of the next form.
2. Ms. Kissell names her table "Colleges" and after thinking through the types of queries she wants to enable, she specifies that it will have 5 columns.
   This information is also saved in variables and used to generate a custom version of the next form. Note that the authoring tool also provides FAQs and other help information to guide project planning and specification.
3. The next form is extensive; for each of the 5 requested columns Ms. Kissell enters a name and a description. The names will be used as column headings, as well as fields that can be queried. The descriptions are used to prompt input during project creation. Ms. Kissell specifies Name, City, State, Tuition, Avg SAT as column names; she then presses the Save button.
   When the project is saved, three tables are populated in the MySQL database. One holds Project information, e.g. name, description and author. A second holds Table information, e.g. table name, description and the project it serves. A third holds Column information, e.g., column name, description and the table and project it serves.

Once a wProjects specification is saved, it can be instantiated by a student. When this happens, the specification content is mapped into variables, and used to generate the custom forms presented to the student. The student is guided to add data records, and this information is stored in a fourth table in the MySQL database; each record holds student data along with column and table identifiers. A fifth table stores the parameters of any queries a student creates. A special web page editing function supports insertion of saved queries as part of a web page. Other functions (most inherited from the Drupal module we adapted) support simple page editing (e.g., fonts, color, images) as well as selection from among a number of pre-set visual themes.

2.3 Design Tradeoffs

The wConnect Web Projects tool addresses many issues we faced with Bridgetools: anyone can use the tool, from anywhere and without preliminary configuration; the web form interface is simple and familiar; the system can be maintained and refined by wConnect community members. It also adds a new layer of functionality via the designer role, wherein teachers or mentors can specify their own project ideas.

At the same time, wProjects introduces new downsides. For instance, though web forms are familiar to Internet users, they can be tedious to view and use (e.g., relative to WYSIWYG object editors in Bridgetools). The Drupal architecture enforces many constraints, for instance the use of themes, the relationship of forms and pages. We are also now linked to Drupal's OSS process, so as new versions are released, our members must monitor this process and learn new skills for release testing. Finally, the database manager within Drupal is less flexible than our Java tools, making the use of queries less dynamic (e.g., revised queries must be re-saved before pages update). We are exploring workarounds for such issues while also beginning to gather empirical data concerning their consequences for designers and learners.

3 Status and Open Issues

Our initial exploration of Drupal as a platform for end-user web application authoring has been promising. However many issues remain. We have already begun empirical evaluation of wProjects, both to refine it and to compare it to our experiences with Bridgetools. One key issue that has already emerged is the availability of examples for teachers that convey meaningful teaching goals; Wiedenbeck [10] has reported similar issues in her observations of teachers as end-user programmers. In parallel we will continue to test the boundaries of what we can accomplish within the CMS. We will explore options for dynamic query and page rendering, and tracking of projects over time. Eventually we will integrate this tool within the wConnect online community, so that any member might choose to become a project designer or builder.

References

1. Whyte, W.F.: Advancing scientific knowledge through participatory action research. Sociological Forum 4(3), 367–385 (1989)
2. Leonard, E.B.: Women, Technology, and the Myth of Progress. Prentice-Hall, NY (2003)
3. Margolis, J., Fisher, A.: Unlocking the Clubhouse: Women in Computing. MIT Press, Cambridge (2002)
4. Rosson, M.B., Sinha, H., Zhao, D., Carroll, J.M.: wConnect: A developmental community for women in computer and information science (2010), `http://www.intechweb.org`
5. Rosson, M.B., Carroll, J.M., Zhao, D., Paone, T.: wConnect: A Facebook-based developmental learning community to support women in information technology. In: Proceedings of Communities & Technology 2009, pp. 125–134. ACM, New York (2009)
6. Rosson, M.B., Ioujanina, A., et al.: A scaffolded introduction to dynamic website development for female high school students. In: SIGCSE 2009, pp. 226–230. ACM, New York (2009)
7. Rosson, M.B., Carroll, J.M.: Developmental learning communities. Journal of Community Informatics 2(2) (2006)
8. Rosson, M.B., Thiry, E., Zhao, D., Carroll, J.M.: Developing an online community for women in computer and information sciences: A design rationale analysis. In: HICSS 44 (2011)
9. Rosson, M.B., Sinha, H., Hansford, T., Mahar, J.: Offering early success expe-riences in software construction. In: ICALT 2008, pp. 458–460. IEEE, Washington, DC (2010)
10. Wiedenbeck, S.: Facilitators and inhibitors of end-user development by teachers in a school. In: Proceedings of VL/HCC 2005, pp. 215–222. IEEE, Los Alamitos (2005)

A Framework for User-Tailored City Exploration

Jacqueline Floch

SINTEF ICT
NO-7465 Trondheim, Norway
jacqueline.floch@sintef.no

Abstract. Despite advances in mobile computing technologies, end-user development has mainly focused on desktop applications. Fundamentally contrasting desktop applications, mobile applications should satisfy the changing needs and tasks of the users on the move, and usually have a strong affinity with the physical world surrounding the users. We illustrate the potentials of this novel area for end-user development through the case of city exploration and we discuss how the different techniques of tailoring, sharing of user-generated content and code, and service composition can be exploited by users to create a tailored city exploration.

1 Introduction

Mobile computing has the potential to support users in most everyday tasks, including their activities at home, at work, as well as on the move. Unlike desktop software that assumes a static view of the user needs and operating conditions, future mobile application scenarios underline the importance of making interaction with devices as transparent as possible but yet effective, maximizing the utility perceived by the users [1]. While this need for transparency has lead to a significant amount of research on context-awareness and self-adaptation of mobile systems [2,3], little work has been done on empowering users to develop themselves applications adapted to their needs and tasks in a mobile context. Truly, developing mobile software has until now been complex –also for professional developers. As mobile devices are becoming more powerful in terms of computing resources and interfaces, we expect mobile software development to get more accessible for all. New platforms and development environments, such as provided by Apple and Android, also facilitate and popularize the development of mobile applications and their distribution. However, these environments target expert developers and do not address the needs of ordinary users.

As a starting point for understanding the challenges and potentials of end-user development in the area of mobile computing, we have selected city exploration as a focus application area. Obviously, city exploration is inherently mobile. Moreover, it is applicable in different domains, such as tourism, games and e-learning. This diversity of domains leads to varying requirements to city exploration and associated mobile applications. Further, earlier research in the tourism application domain has shown positive acceptance of mobile guiding, and provides a number of valuable recommendations for the design of an initial

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 239–244, 2011.

city exploration framework. In particular, it shows that a tourist guide should accommodate a wide range of user needs and let users to choose the level of functionality that they require [4].

This paper presents our initial results. We first position our approach to related work (Section 2) and discuss our research method (Section 3). Then starting from a set of scenarios, we seek to identify a set of software services relevant for city exploration and to understand how particular user situations and needs influence the configuration and composition of these services (Section 4). We also introduce the initial design of a concrete city exploration framework that combines simple tailoring, sharing of user-generated content and code, and service composition as means for the user to adapt city exploration to his particular needs (Section 5).

2 Related Work

Little research has been dedicated to the end-user development of mobile applications. This is not surprising as mobile devices had until recently limited capabilities and supported few advanced functionalities. Rather approaches for defining user profiles and triggering device components have been proposed. For instance, in [5] a tool is described that support users in specifying context-action rules for customizing the triggering of components (e.g. send SMS, set up call, etc.) to context (e.g. location, noise, device activity) on the Nokia Symbian Series 60 mobile phone. Exploiting more advanced platform support, PdaGraph [6] is a data-flow visual language for end-user component-based programming on PDA. Since this research was conducted, the premises for end-user programming on mobile devices have tremendously changed. On one hand, a large number of users are familiar with smartphones today. On the other hand, new interface technologies greatly increase the usability of devices and applications, and provide new concepts for the design of end-user tools.

The App Inventor framework [7] that was proposed by Google Labs early summer 2010, supports the development of applications for the Android platform. Differently from the two approaches described above, programming is performed on a desktop –not on the mobile device itself, and, after development, the resulting application is downloaded to the mobile device. While App Inventor targets the development of generic applications, we seek to understand the potentials and requirements of the end-user development in the area of mobile computing through a specific application case.

As relevant application domains for city exploration are concerned, mobile and context-aware city guiding was a hot research topic in the 90's. The research was motivated by the exploration of an emerging mobile technology [4,8]. Commercial applications currently available on smartphones, e.g. LonelyPlanet and Beeloop on iPhone, do not exploit earlier research results. They are typically are structured as books and, besides navigation support, do not take advantage of other services made available through the mobile platform, such as on-line reservation services. They do not either provide support for tailoring to individual needs, such as itinerary creation. Further, commercial applications are bound to large cities; none is available for medium-size and small cities. Targeting museums rather than cities, Cicero Designer [9] supports end-user creation of museum guides. By letting the creator attach quizzes or other games to museum artefacts, this tool exemplifies that exploration is not restricted to providing information. Differently from our framework, using Cicero Designer, the guides are created on a desktop platform. Beyond the domain of tourism, city exploration has also been applied in the context of e-learning. In [10], a platform facilitating the development of interactive city games is proposed. The intention is to let pupils get known with the cultural city through exploration and collaboration. This work does not address end-user development. Our ambition is to let teachers compose and configure the exploration application used by the pupils.

3 Research Approach

Our research follows the design science methodology [11] where development and improvement of software artefacts is central for understanding a research problem and answering questions related to the problem. A software artefact can be a number of different things, such as scenarios illustrating software uses, specifications and implementations. In this paper, we address two main questions: 1) What mobile services are relevant for city exploration, and how do different users' needs influence the configuration and composition of these services? 2) What mechanisms can be exploited to configure and compose these services on a mobile platform?

To address the first question, a set of scenarios involving different kinds of users and illustrating different user activities were created. The scenarios were first sketched by IT-researchers and -students. The reason for that choice is twofold. First, end-user development of mobile applications is a novel research area that we do not understand well and therefore find difficult to communicate to end-users. Second, few end-users understand what can be achieved through exploiting the flexibility of an emerging mobile technology as there is no precedence in the field. Focus was set on the tourist domain that all contributors to the scenarios had a good knowledge of. After an initial sketching, scenarios were evaluated and improved through interviews of tourism professionals at two tourist offices in Norway.

To address the second question, an initial design of the city exploration framework was specified, and the resulting specifications presented and evaluated by end users. Due to space limitations, we do not present the details of this initial design in this paper, but rather focus on important design decisions. The framework is currently under realisation aiming at user evaluation of the prototype.

4 Scenarios for Tailored City Exploration

The scenarios describe various activities during city exploration and depict the mobile services used to perform these activities. Each scenario consists of a main story and a set of alternative behaviours corresponding to different user needs. In that way, the scenarios also explore how and when end-user development can be applied in the setting of city exploration. In the following, we provide a summary of the main services identified through the scenario work. A complete description of the scenarios is available at [12].

The scenarios cover different phases of a city exploration: planning a visit, performing the visit and reporting experiences after the visit. In all phases, we can distinguish between domain-specific services that most users probably expect from a city guide and user-specific services that relate to individual needs under a visit. While the latter is most relevant from the viewpoint of end-user development, the former is also interesting as the services illustrate a need for tailoring that, today, is not supported by commercial applications. Tailoring applies both to the visit and the level of functionality supported by the guide. We aim at taking advantage of the combination of tailoring of domain-specific services and creating user-specific services when introducing end-users to the city exploration framework. Indeed this facilitates an incremental development and a gradual acquisition of skills –with tailoring first and creation of user-specific services later, as recommended in early research on end-user development [13].

Tailorable domain-specific services include:

- retrieval of site descriptions, related services and itinerary suggestions offered by different actors such the tourist office or cultural associations;
- selection of favourite sites and creation of tailored itineraries possibly setting up fixed route;
- inclusion of city-related software services, i.e. transport or parking services;
- navigation support and access to online-guiding services available at different site;
- discovery and recommendation of new sites on the fly;
- update of itineraries on the fly with inclusion of new sites or postponing of visits;
- information sharing on a blog and reviewing of sites;
- annotation of sites with pictures and possibly route recording for later picture tagging and sharing with friends.

In addition to these domain-specific services, the scenarios depict user-specific services such as:

- definition of triggers for the activation of specific software services with/without involvement of the user, e.g. looking for a parking spot when getting close to the city, getting bus routes to the next place in a tailored itinerary when approaching the end of a visit, retrieving site description when arriving to a site, or posting information on a blog when arriving to a site;
- support for group collaboration, e.g. positioning group members according to places in itineraries, or setting up meeting points with notification to the group members;
- definition of areas where a visitor should stay, e.g. kids should stay in areas as agreed with their parents and if not notifications should be sent to parents.

5 Initial Design of the City Exploration Framework

This section introduces the initial design of the city exploration framework. As shown in Fig. 1, the framework supports four main functions: 1) Sharing of artefacts such as descriptions of sites, games and other software related to sites, itineraries linking sites together and services composed by the users; 2) Tailoring of domain-specific services whereusers configure and set up own visits through the selection of favourite places and creation of itineraries; 3) Composition of user-specific services (Note that the analysis of the scenarios does not depict the need to support detailed programming. Rather the end-users need to describe the activation of composite services in the case some events occur.); 4) Exploration of the city using the tailored and composed services.

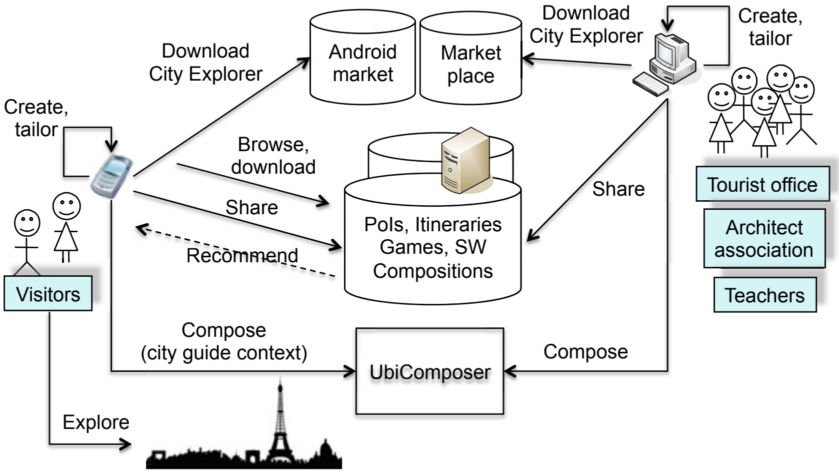


Fig. 1. Overall picture of the city exploration framework

The users of the framework play different roles, as providers or consumers of the shared artefacts. For instance, tourist offices and associations typically provide information, while visitors consume information. Depending of the situation of use, the framework should be available on a desktop or a mobile platform. In a first time, we concentrate on the realisation on a mobile platform. Android was chosen because of its support for flexible software composition and its availability as open source software.

We propose to address domain-specific tailoring and composition of user-specific services separately and introduce two applications: City Explorer for sharing, tailoring and exploration, and UbiComposer for the composition of user-specific services. This is done with consideration of reuse as we observe that the user-services depicted in our scenarios can also be used in other domains. For instance, uploading annotated pictures is useful in an accident situation, and person tracking is relevant for caregivers of persons mentally challenged.

An initial exploratory study with interviews of test users was conducted to find out 1) how the concepts were perceived by ordinary users, 2) the willingness to use such a framework. 12 frequent travellers from all over the world, women and men, aged between 25 and 60, and with various education background were recruited to take part in the study. All had experience with using smartphones. Our results show that the participants are favourable to the concept of tailoring and feel confident they could use it. The concept of composition gets lower scores, but still the results are positive and encourage us to further develop the concept. We also observe that the youngest participants (under 35) were in general more favourable to the concept.

Acknowledgement. Our research has been performed in the Norwegian project UbiCompForAll.

References

1. Rodden, T., Cheverst, K., Davies, N., Dix, A.: Exploiting context in HCI design for Mobile Systems. In: Proc. of Workshop on Human Computer Interaction with Mobile Devices (1998), http://eprints.lancs.ac.uk/id/eprint/11619
2. Baldauf, M., Dustdar, S., Rosenberg, F.: A Survey on Context-Aware Systems. Journal of Ad Hoc and Ubiquitous Computing 2(4) (2007)
3. McKinley, P.K., Sadjadi, S.M., Kasten, E.P., Cheng, B.H.C.: Composing Adaptive Software. IEEE Computer 37(7) (2004)
4. Cheverst, K., et al.: Developing a context-aware electronic tourist guide: some issues and experiences. In: Proc. of SIGCHI Conference on Human Factors in Computing Systems (CHI). ACM Press, New York (2000)
5. Häkkilä, J., Korpipää, P., Ronkainen, S., Tuomela, U.: Interaction and End-User Programming with a Context-Aware Mobile Application. In: Costabile, M.F., Paternó, F. (eds.) INTERACT 2005. LNCS, vol. 3585, pp. 927–937. Springer, Heidelberg (2005)
6. Kollet, Y., Smedley, T.J.: Message-Flow Programming in PdaGraph. In: Proc. of IEEE Symposium on Visual Languages and Human-Centric Computing (VLHCC). IEEE, Los Alamitos (2004)
7. App Inventor for Android, http://appinventor.googlelabs.com/about/
8. Abowd, G., et al.: Cyberguide: A mobile context?aware tour guide. Wireless Networks 3(5) (1997)
9. Ghiani, G., Paternò, F., Spano, L.D.: Cicero Designer: An Environment for End-User Development of Multi-Device Museum Guides. In: Pipek, V., Rosson, M.B., de Ruyter, B., Wulf, V. (eds.) IS-EUD 2009. LNCS, vol. 5435, pp. 265–274. Springer, Heidelberg (2009)
10. Kathayat, S.B., Bræk, R.: Platform Support for Situated Collaborative Learning. In: Proc. of Int. Conf. on Mobile, Hybrid, and On-line Learning (eL&mL). IEEE Press, Los Alamitos (2009)
11. Hevner, A.R., March, S.T., Jinsoo, P.: Design Science in Information Systems Research. MIS Quarterly 28, 75–105 (2004)
12. UbiCompForAll, Project Home Page, http://www.ubicompforall.org
13. Repenning, A., Ioannidou, A.i.: What makes end-user development tick? 13 design guidelines. In: Lieberman, H., Paterno, F., Wulf, V. (eds.) End-User Development. Springer, Heidelberg (2006) ISBN 1-4020-4220-5

End-User Requirements for Wisdom-Aware EUD

Antonella De Angeli, Alberto Battocchi, Soudip Roy Chowdhury,
Carlos Rodriguez, Florian Daniel, and Fabio Casati

Department of Information Engineering and Computer Science
University of Trento
Via Sommarive, 14 – 38123 Povo, Trento (Italy)
{antonella.deangeli,alberto.battocchi,soudip.roychowdhury,
carlos.rodriguez,florian.daniel,fabio.casati}@unitn.it

Abstract. This paper presents requirements elicitation study for a EUD tool for composing service-based applications. WIRE aims at enabling EUD by harvesting and recommending community composition knowledge (the wisdom), thus facilitating knowledge transfer from developers to end-users. The idea was evaluated with 10 contextual interviews to accountants, eliciting a rich set of information, which can lead to requirements for Wisdom-Aware EUD.

1 Introduction

There are two common approaches to enable less skilled users to develop software artifacts. Development can be eased by simplifying it or by reusing knowledge. Among the *simplification approaches*, the business process management and service computing communities have focused on abstracting process development and service composition into activities, as well as control and data flows. However, these are still challenging tasks even for expert developers [1,2]. Traditional *reuse approaches*, in the form of program libraries, services, or templates (such as generics in Java or process templates in workflows) have targeted developers rather than end-users. Recently, some effort has been invested into knowledge reuse techniques for end-users. In programming by demonstration [3], the system auto-completes a process definition, starting from a set of examples chosen by the user. Goal-oriented approaches [5] assist the users by automatically composing solutions that satisfy user-specified goals. Pattern-based development [4] proposes the use of libraries of patterns provided by experts to represent good development practices, yet patterns, such as the glue patterns in [7], may also be derived from existing compositions. Syntactic approaches [11], for instance, suggest operators based on syntactic similarity (comparing output and input data types), while semantic-based approaches [6] annotate ingredients to support the retrieval of semantically matching elements.

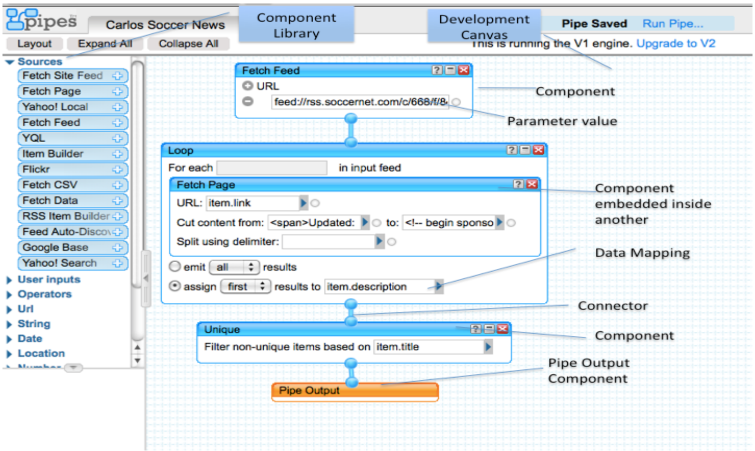
While some of these approaches support end-users with reusable knowledge, they all suffer from some shortcomings. Programming by demonstration and goal-based approaches propose "best", complete solutions, not allowing the user to control which exact ingredients the solution should contain. Pattern- and semantics-based approaches are hard to maintain, in that they require explicit input from human experts.

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 245–250, 2011.

In this paper we present the results of a requirement study for *WIRE* (*WIsdom-awaRE development environment)* a EUD tool to exploit the benefits of simplification *and* reuse. WIRE targets process-oriented, mashup-like applications that are characterized by relatively simple composition logics and complex tasks or components. This class of programs provides the benefit of simplicity (composition, not coding) and a sufficient information base (the compositions themselves). The idea is to *learn from existing compositions* developed by expert IT developers and provide learned knowledge in the form of interactive recommendations to facilitate EUD.

2 WIRE

The motivation behind the idea of WIRE has derived by the analysis of the shortcomings of existing mashup development tools. To exemplify this claim, let us consider a simple application created by Yahoo! Pipes, which retrieves news feeds from a specified website, filters the content based on user-specified criteria, and publishes the filtered content for viewing (Fig. 1). Such a simple application requires 5 components. The user has to set the value of the configuration parameters of a component (e.g., the *URL* Parameter of the *Fetch Feed* component) and define the data-flow logic between components. Assuming that an end-user has this kind of technical knowledge is not realistic.



URL: feed://rss.soccernet.com/c/668/f/8

For each in input feed

Fetch Page

URL: item.link

Cut content from: <span>Updated: to: <!-- begin sponso

Split using delimiter:

○ emit all results

● assign first results to item.description

Filter non-unique items based on item.title

Fig. 1. Implementation of the example scenario in Yahoo! Pipes

WIRE is aimed at discovering technical knowledge by analyzing existing, successful applications, storing knowledge as development advice (“*community composition knowledge”*[8]), and delivering it in the form of contextual *interactive recommendations* to the end-user. The intuition is that this knowledge can be captured through *composition patterns* and reused as recommendations. The patterns we identified include Parameter Values (e.g., values for the *URL* parameter in the *Fetch Feed* component), Component Associations (e.g., suggest that a *Loop* component should be added together with a *Fetch Feed* component), Connectors (e.g., possible connections between components), Data Mapping (e.g., suggest that the *item.link* element coming from the *Fetch Feed* component should be mapped to the *URL* parameter of the *Fetch Page* component), or Complex patterns (e.g., suggest adding components based on a Component Association pattern together with the wiring among them based on a Connector pattern). A detailed explanation of the conceptual model and architecture of WIRE is presented in [8].

3 User Study

An evaluation of the conceptual design of WIRE was run in order to address benefits and limitations of the proposal and elicit user requirements [12]. The evaluation was based on contextual interviews to 10 University accountants (7 F, 3 M; mean age = 37 years of age), which lasted approximately one hour. None of them had a background in computer science. Participation was rewarded with 15 Euros. The interview addressed two main topics. Section A targeted the strategies that people use for overcoming the difficulties that emerge while using computers during day-to-day work, and their attitudes towards computer-provided help and advice with particular focus on the comparison between automatic/contextual and on-demand help. Participants were shown a slideshow of familiar examples of automatic/contextual advice (i.e., *word completion* in the Google search box, *friend suggestion* in Facebook, *book suggestions* in Amazon, *passwords auto-save* in web-browsers, *pop-up reminder* on calendars, *related videos* sidebar on YouTube), invited to comment on each example, and report their understanding on how the advice was created.

Section B collected opinions and suggestions about WIRE by a plus and a minus scenario [9] reporting on an accountant who is using WIRE for automating the process of management of travel reimbursement. Both scenarios described the effects brought forward by WIRE on a new user. These effects were taken to the positive or negative extreme to help users to think what consequences the approach could have in their work practices. In the *Positive Scenario*, the accountant had a successful experience, which helped him to save time and speed up repetitive work leading to adoption. In the *Negative Scenario*, the accountant encountered serious difficulties and eventually decided to go back to his traditional work procedures. Scenarios were presented with a counterbalanced order. Interviewees were asked specific questions addressing their willingness to use the system, advantages and drawbacks, preference for contextual or on-request help, and for the way the help was presented.

4 Results

Asking to colleagues and technicians represented the most common option used by half of the interviewees to seek for help and advice. The person to whom they asked for help was usually chosen on the basis of his/her level of expertise or on friendship/acquaintanceship. Google represented the first choice of help for four of the participants and the second choice for those participants who could not find a solution to their problems asking colleagues or technicians. Participants reported using online help and help menus rarely, and this was the first choice only for one interviewee. When asked which source of help was the most effective, eight participants indicated colleagues and technicians. Their choice was motivated by the fact that technicians are professional and helpful, and that providing support is part of their job. One participant indicated Google as the best source of information *"because you can use it at any time, also when you are at home"* (P10).

Seven participants reported a preference for automatic/contextual help rather than help on-request, but two of them also specified that this method works better for new or simple applications. Participants suggested that the automatic/help function should be customizable in order to be really useful. One participant provided an interesting observation about the function of automatic/contextual help:

> *"Automatic/contextual help has a double function: it appears when you need help and reminds you of potential errors; help on request covers only the first function"* (P10).

Participants provided valuable comments on the effectiveness and usefulness of common examples of contextual advice. People favoured less intrusive contextual advice, that do not try to guess the user's preferences or opinions, and that do not present risks for data security, such as Google's *automatic word completion*, *pop-up reminders* in Google Calendar, and the *related videos sidebar* in YouTube. Contextual advice was valued mainly in the case of "objective" suggestions (e.g., YouTube) but perceived as less accurate when it tries to enter users' private space (e.g., Facebook). When asked to formulate their "naïve theories" about contextual help is generated, all the participants reported that they are created on the basis of the inserted keywords. One participant also made a distinction between general, or simple, and particular, or complex, suggestions:

> *"For simple queries, the system works on simple analogies with the inserted keywords; for more complex issues, the system does a matching with your personal characteristics (provided while registering to a service"* (P8).

Participants provided useful information about their attitude toward WIRE. When reading the positive scenario, participants recognized several similarities with their work practices and perceived the system as potentially very useful. Two participants expressed a common concern about the introduction of WIRE into their work practices and suggested that, in order to benefit of its potentialities, the use of WIRE should totally replace previous practices, without leaving space for overlapping. The Negative Scenario was also perceived as very plausible as it described well fears and frustrations that may emerge when something goes wrong dealing with new systems or procedures. In particular the interviewees stressed the need for a system which is well designed and thoroughly tested before being introduced into the work practice:

> *"I gave for granted that this technology was previously tested and approved by the central administration office. [...]. In the case of dealing with sensitive or financial matters, I would trust the system only if I am 100% sure that it is effective and functional"* (P7).

Participants were asked if they would be interested in using WIRE. Nine of the interviewees responded positively and one was openly sceptical stating that *"using WIRE would take the same time it takes doing the procedure manually"* (P1). Anyway, formal training was indicated by two participants as a fundamental prerequisite for adoption. Drivers to adoption were identified in *better organization of work*, *optimization of time*, *reduction of errors*, and *sharing of procedures and methodologies with colleagues*. Major obstacles were connected to *loss of control over work processes*, in the case that these were entirely completed in an automatic way:

> *"I would like to keep track of each step of the process; if everything is made automatically, the users misses the logic that stays behind the process"* (P4).

All the participants declared that the advice provided by WIRE in the scenarios would be effective in meeting their needs, as they were generated on the basis of past experience of colleagues that share the same work procedures and possibly the same difficulties. Interviewees showed a marked preference for contextual help (8 participants) over help on-request; two of them added that the possibility of personalizing the way suggestions are provided would be a very important feature in order to make help messages really effective. Help messages provided *during the task* were preferred to messages provided before the task by nine of the interviewees. One participant suggested that the two modalities could be combined:

> *"I can see the two modalities as complementary. At the beginning of the activity the system asks what your needs are in general; during the activity, pop-up windows provide you solutions when the system feels that you are stuck"* (P8).

5 Conclusion

End-users acknowledged that the idea of WIRE for providing assistance, which was derived from the experience of colleagues working in a similar context, was useful. However, issues related to *trust*, *timing* and *usefulness* of the advice still remained as concerns to the users. During the design of WIRE we will need to find new strategies to make its' operations transparent e.g. showing users how the advice is generated and why a particular advice is suggested in a given context. Transparency will also help to build up the trust of the users to use a recommendation tool like WIRE. This is particularly important when people deal with the sensitive and financial issues. *Personalization* is another desired feature, which enables users to receive optimized advice based upon their expertise level. Helping users with the personalized advice can certainly reduce the barrier for adopting this kind of EUD tools to a larger end-user community.

The study provides support to the proposal of collaborative tailoring, discussed in [10], as often participants mentioned that their willingness to engage in EUD was mediated by having support from other people and technical help easily available to them. This help was meant not only to alleviate some of the technical difficulties they had to face during development but also to take the responsibility out of their hands, making them less accountable in case of software failures. Issues related to organizational regulations and corporate processes also emerged as barriers to general EUD uptake, as people often mentioned the need to have explicit approval from their manager as a fundamental step towards making them willing to explore new techniques and tools to automatise their work practices.

Acknowledgments. This work was partially supported by funds from the European Commission (project OMELETTE, contract no. 257635; project SERVFACE, contract no. 216699).

References

1. Namoun, A., Nestler, T., De Angeli, A.: Conceptual and Usability Issues in the Composable Web of Software Services. In: Daniel, F., Facca, F.M. (eds.) ICWE 2010. LNCS, vol. 6385, pp. 396–407. Springer, Heidelberg (2010)
2. Namoun, A., Nestler, T., De Angeli, A.: Service Composition for Non Programmers: Prospects, Problems, and Design Recommendations. In: ECOWS (2010)
3. Cypher, I., Halbert, D.C., Kurlander, D., Lieberman, H., Maulsby, D., Myers, B.A., Turransky, A. (eds.): Watch what I do: Programming by Demonstration. MIT Press, Cambridge (1993)
4. Van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow Patterns. Distributed and Parallel Databases 14(3), 5–51 (2003)
5. Henneberger, M., Heinrich, B., Lautenbacher, F., Bauer, B.: Semantic-based Planning Process Models. In: MKWI 2008, Munich, Germany (2008)
6. Ngu, A., Carlson, M., Sheng, Q., Paik, H.: Semantic-Based Mashup of Composite Applications. IEEE Transactions on Services Computing 3(1), 2–15 (2010)
7. Greenshpan, O., Milo, T., Polyzotis, N.: Autocompletion for Mashups. In: Proc. VLDB Endow., vol. 2(1), pp. 538–549 (2009)
8. Roy Chowdhury, S., Rodríguez, C., Daniel, F., Casati, F.: Wisdom-Aware Computing: On the Interactive Recommendation of Composition Knowledge. In: Proc. of WESOA (2010)
9. Bødker, S.: Scenarios in user-centred design - setting the stage for reflection and action. Interacting with Computers 13, 61–75 (2000)
10. Wulf, V., Pipek, V., Won, M.: Component-based tailorability: enabling highly flexible software applications. Int. Journal of Human-Computer Studies 66(1), 1–22 (2008)
11. Wong, J., Hong, J.I.: Making mashups with marmite: towards end-user programming for the web. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2007), pp. 1435–1444. ACM, New York (2007)
12. De Angeli, A., Battocchi, A., Roy Chowdhury, S., Rodriguez, C., Daniel, F., Casati, F.: Conceptual Design and Evaluation of WIRE: A Wisdom-Aware EUD Tool. Technical Report DISI-11-353, Ingegneria e Scienza dell'Informazione, University of Trento

Personalised Resource Categorisation Using Euler Diagrams

Paolo Bottoni[1], Gennaro Cordasco[2], Rosario De Chiara[2],
Andrew Fish[3], and Vittorio Scarano[2]

[1] Department of Computer Science, “Sapienza” University of Rome - Italy
[2] ISISLab - Dipartimento di Informatica, Università di Salerno - Italy
[3] School of Computing, Engineering and Mathematics, University of Brighton - UK

Abstract. The categorisation of information is a very common practice. Often, the user may need to use multiple hierarchies or require multiple characterisations to be active at the same time, or they may wish to define cross-cutting groups for special purposes. The work developed in this paper is aimed at providing a flexible, seamless management of various ways of organising the required information. The concept of a set is adopted as the fundamental notion of information organisation, and, in particular, the familiar visual representation of sets and their relationships in terms of Euler Diagrams is used. We facilitate the visualisation of sets, enabling the application of functions to items presented in regions of the diagram corresponding to set, or category, intersections. We present a system that realises this novel concept, together with rationale for the choices made in its development, as well as a simple scenario of use.

1 Introduction

Users continually make categorisations in their daily practices and exploit them to perform personal or organisational tasks. They often need multiple hierarchies or multiple characterisations to be active at the same time, or to define cross-cutting groups for special purposes. In such situations, current tools either require the use of complex procedures, or do not allow such usages at all.

Typically, the main categorisation provided by current file systems is an inflexible one, and hierarchical-based representations have well-known limitations in supporting user categorisation. On the other hand, the explicit construction of virtual folders is usually restricted to specific cases (e.g. emails), that are typically used to save the results of some search. As a result of this, users may have to deal with two different types of categorisation, which potentially may cause cognitive difficulties for non-sophisticated users in particular. Therefore, helping users to efficiently define and maintain the desired categorisations in the context of resource-based tasks which are routinely performed, or even for special purposes, would be of benefit in such situations.

The tagging of documents, or the use of metadata in general, is an emerging methodology for handling resources, but tag-based methodologies have the downside that documents with missing or mistyped tags may be omitted, and they can run into trouble if one also wishes to utilise methods based on file types. Although modern operating

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 251–257, 2011.

systems support both tag and file-type based save, search and retrieval methods, they do not explicitly exploit spatial features, leveraging users' spatial abilities and memory.

The work developed in this paper is aimed at providing a flexible and seamless management of different ways of organising information, whilst exploiting visual representations to facilitate user understanding of the structure of the information. Through these representations, users should be able to dynamically define the categories they are interested in, to manage multiple characterisations at the same time, and to access or define actions relevant to the different categories. In particular, we propose to adopt sets, rather than folders, as the fundamental notion of information organisation, and to exploit their familiar visual representation in terms of Euler Diagrams (EDs).

The main contribution of the paper is the realisation as a flexible framework, together with an implementation of a user interface for the construction and modification of Euler Diagrams as a non-hierarchical categorisation structure. We also provide a palette of operations, permitting a user to select from a collection of predefined operations and to apply the operations to the zones of an ED. The interface permits end users to develop their own functionalities or to combine functionalities into macros.

Related Work. The categorization process can be viewed as the process of assigning tags to items. Hierarchies have been so widespread since the advent of personal computing that they are often automatically considered the natural way to organize data. There are many different techniques to visualize hierarchies (cf. [13,14,15]). Since hierarchical classification structures are often not sufficient for user classification needs, one can consider non-hierarchical classification structures, such as polyarchies [17] or EulerView [4].

Venn diagrams were used to represent non-hierarchical directories, replacing the traditional structure of file systems in [3], where diagrams could be drawn with curves representing categories (or tags) and files could be placed within more than one directory by utilising curve overlapping. In [6], an accessible Euler diagram interface was developed which enabled more general resource management, together with efficient interpretation algorithms to detect the underlying meaning of the regions of the diagram. A reification of an ED-based categorisation structure was integrated into Flickr in [5], utilising the non-hierarchical categorisation structure.

There has been relatively little user testing of the ED concept, with the idea generally being taken for granted as being beneficial. In [2], the comprehension of basic EDs (without items) with the same zone sets, but different visual properties e.g. curve jaggedness, was examined. In [8] the effects on user comprehension and preference of varying well-formedness conditions were investigated.

2 Background on Euler Diagrams

An *Euler diagram* is a pair $d = \langle \mathcal{C}, \mathcal{Z} \rangle$ where $\mathcal{C} = \mathcal{C}(d)$ is a set of labelled *contours* (closed curves) in the plane and $\mathcal{Z} = \mathcal{Z}(d)$ is the collection of *zones*, z, which are specified as being inside a set of contours, $X_z \subseteq \mathcal{C}(d)$, and outside the rest of the contours of the diagram; X_z is called a *zone descriptor* (for details see [9]).

Figure 1 shows an Euler diagram with four labelled curves representing the categories: $music$, $technology$, $work$, $gadget$. This set of curves decomposes the plane

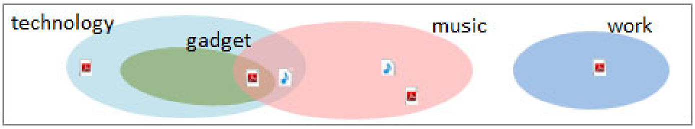


Fig. 1. An example of an ED used for resource categorisation

into a set of seven zones (the region outside all curves is itself a zone). As per the notion of zone descriptor, we describe zones by specifying the labels of the complete set of curves that the zone is inside, with the outside set of curves being derivable from this information. For instance, Figure 1 presents two music files (making use of the icons), one belonging to the zone $\{music\}$ and the other to the zone $\{music, technology\}$.

We view the zones of an Euler diagram as a repository in which to place resources, such as `filess` and `urls`, and so the basic notion of Euler diagrams is extended to capture the placement of items in the diagram. We assign to each zone, z, the set of tags in its zone descriptor, X_z, and extend this assignation to any items placed in that zone. Thus an ED provides a means to build a non-hierarchical categorisation structure, and can be used to categorise resources, as in [6].

3 Categorising Items with FunEuler

We describe the FunEuler system, together with the rationale for the design decisions adopted. The application is available at [1].

Architecture. FunEuler is conceived as a simple application running beside the windows that a user uses to manipulate their files and documents. FunEuler is implemented in Java on top of the EulerVC library [7]. This library provides a number of set manipulation functions and it enables the interactive creation and modification (e.g. curve addition, removal, or transformation) of an ED. It also allows the manipulation of items, which are simple elements that associate a resource to a two dimensional co-ordinate.

FunEuler is filetype agnostic, handling the placement (via drag and drop) of resources from any repository of files as well as URLs. An example of the flexibility of the paradigm is that it is permissible to categorise files (e.g. .doc, .pdf) and URLs (e.g. email addresses and bookmarks) using the same ED. We highlight that data in FunEuler is not duplicated, with FunEuler keeping track of both the resource location in the repository and the position in which the user placed the files on the ED.

Figure 2 shows the functional blocks of FunEuler: 1) repositories from which the items to be categorised can be retrieved (left); 2) a library of functions, see [7], for diagram manipulation and query operations for zones and items (centre); 3) processes for invoking operations (right), where arrows indicate the parameters to be passed.

Figure 3 shows a screenshot of the current user interface, comprised of three panes: the palette of operations, the Euler Diagram and the Results Euler Diagram. The left-hand pane contains a list of icons for the possible operations that the user may apply to the regions of the ED shown in the main pane. The main portion of the application window contains the ED that is used for categorisation and this is where the majority of

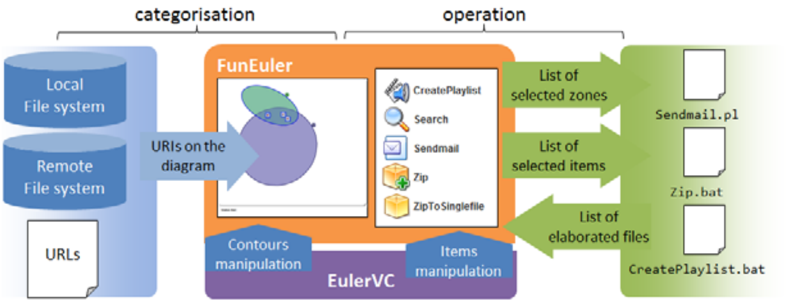
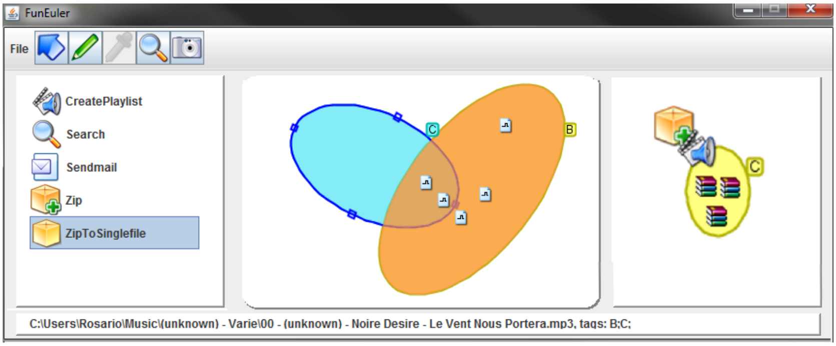


Fig. 2. FunEuler architecture



C:\Users\Rosario\Music\(unknown) - Varie\00 - (unknown) - Noire Desire - Le Vent Nous Portera.mp3, tags: B;C;

Fig. 3. A screenshot of the FunEuler interface

the user-interaction is likely to occur. The rightmost pane shows a diagram that depicts the output from the application of the user-selected operations.

Diagram construction and interaction. Users can construct a diagram by adding curves, in the form of ellipses, one at a time by a simple mechanism of left click and drag to specify one axis, whilst using the mouse scroll wheel for the other axis. The restriction of curves to be ellipses in the EDs does not permit the representation of arbitrary relations between curves, so a user might not be able to draw a diagram containing all and only regions from a specific selection. However, it is important to realise that the creation of a diagram with the exact set of relations is not necessary in this context.

When a user creates a curve c, FunEuler assigns a random colour to c, as well as an automatic choice of set name, according to a pre-determined sequence. These set names, as well as the colours, can then be edited by the user. This enables users to rapidly construct a diagram, via ellipse additions, without having to explicitly name and set a colour for each new curve constructed since this could interrupt the user's reasoning flow. Moreover, every ellipse present in the diagram can be quickly modified by translation, rotation or by shape alteration (i.e. modifying the size of its axes).

In order to try to enhance the ease of comprehension of a diagram, and promote interactivity, FunEuler allows the user to query the diagram: when a user selects a position p in the diagram, exploiting the eyedropper, the entire zone z containing p is highlighted and the zone descriptor X_z is displayed in the status bar. The intent is to simplify item categorization and to help a reader understand an existing categorisation.

To enhance user navigation within larger diagrams, FunEuler provides horizontal and vertical scrollbars together with spatial zooming, via a select/zoom option and scroll in and out, in order to provide a larger working area when needed.

We utilise a drag and drop approach for resource classification: items are placed within the relevant region of the diagram in order to categorise them. Item properties are visualized through tool tip text. The status bar, at the bottom of the interface, shows the current categorization for the selected item. Double click on an item allows the user to access the item using the default program associated with its extension/protocol. Each item can be interactively repositioned and consequently re-classified. Operations applied to single or multiple zones effect all items within those zones. However, in order to not restrict the selection of items (and hence the application of operations) to be all the items within the selected zones, we also allow the selection of multiple individual items that are displayed via a common box-selection.

4 Operational Functionality

At the core of the FunEuler functionalities is the ability to select and apply one or more *operations* to diagram regions. This is performed by using the familiar drag and drop: the user first selects (with the eyedropper) all of the zones to which an operation is to be applied, and then drags the operation icon onto the diagram, placing it on one of the selected zones. Placing the operation icon on a zone which is not already selected applies the operation to that zone, disregarding the selection.

For resources produced as the result of operations performed, we adopt an automatic naming convention of assigning the zone name followed by the type extension. For example, if we were to apply the zip function to the region in the intersection of the "Rock" and "Pop" curves, we would automatically name it `Pop&Rock.zip`.

End User Development. We consider, and encourage, various levels of end user programming or development with this interface. At a very low level, the very essence of the operational application within the interface is allowing users to assemble operations and this could be viewed as building a simple program or macro to support their tasks. However, we consider the development of new operations as being a more sophisticated instance of End User Development and we facilitate this development by offering a simple means to plug in new operations that have been developed or to combine existing operations. This is a key area where we feel the design of the interface will provide benefits. Note that the interface allows End User Programmers to develop operations using their favourite scripting language (e.g. php, python, etc).

To allow the end user to design a new operation, FunEuler provides access to two lists for the items and the zones that the operation is to be applied to: one list containing the pathnames of the items and the zones they belong to, and another list reporting the zones and the items that each zone contains. Each operation can produce zero, one or several files as result. As an example, the `zip file` operation produces a zipped file for each pathname in the input list, whilst the `zip to single file` operation produces one single `zip` file containing all of the files of the input list. On the other hand, the `send` by email operation does not produce any result files. Independent of the number of files produced the Results pane supports their representation in a special single-curve ED, where file icons are automatically placed within the curve representing the results set.

Since the result set is itself an ED, users can apply operations to it. This facilitates the sequential combination of operations, which produces a new Results diagram. The operation icons are added consecutively, on top of the previous ones. At a higher level, we can view the class of diagrams in FunEuler as being *closed under the application of operations*, since the application of an operation to a region of a diagram causes the generation of a new diagram in the Results pane of the application window. Given this closure property, FunEuler allows users to drag the Results diagram onto the main diagram area. Hence, users can categorise items which are the result of one of more operations alongside the original items that were already displayed in their diagram.

5 Conclusion and Future Works

We have brought together concepts of visual classification, spatial arrangements and functional application in a novel Euler diagram based interface called FunEuler.

Future work includes examining how this representation interacts with other natural representations such as geography or time-lines, performing testing on end user programmers (e.g. how difficult do they find concatenation of function, provision of new operations) and extending functionalities to include methods for suggesting synonyms (since not only do different users use different names but that, over time, even the same user uses different names [18]). Scalability is another important issue, and although, computationally the interface and underlying algorithms can deal with large numbers of curves and items, the utility of the interface remains to be tested on users.

References

1. Funeuler homepage, http://www.isislab.it/projects/funeuler
2. Benoy, F., Rodgers, P.: Evaluating the comprehension of Euler diagrams. In: Proc. IV 2007, pp. 771–780 (2007)
3. De Chiara, R., Erra, U., Scarano, V.: VennFS: a Venn Diagram File Manager. In: Proc. IV 2003, pp. 120–126. IEEE CS Press, Los Alamitos (2003)
4. De Chiara, R., Fish, A.: Eulerview: a non-hierarchical visualization component. In: Proc. VLHCC 2007, pp. 145–152. IEEE CS Press, Los Alamitos (2007)
5. De Chiara, R., Fish, A., Ruocco, S.: Eulr: a novel resource tagging facility integrated with Flickr. In: Proc. AVI 2008, pp. 326–330. ACM Press, New York (2008)
6. Cordasco, G., De Chiara, R., Fish, A.: Interactive visual classification with Euler diagrams. In: Proc. VL/HCC 2009, pp. 185–192. IEEE CS Press, Los Alamitos (2009)
7. Cordasco, G., De Chiara, R., Fish, A.: Efficient on-line algorithms for Euler diagram region computation. Computational Geometry 44(1), 52–68 (2011)
8. Fish, A., Khazaei, B., Roast, C.: User Comprehension of Euler Diagrams. To appear in Journal of Visual Languages and Computing (2011)
9. Flower, J., Fish, A., Howse, J.: Euler diagram generation. JVLC 19, 675–694 (2008)
10. Howse, J., Stapleton, G., Taylor, J.: Spider diagrams. LMS Journal of Computation and Mathematics 8, 145–194 (2005)

11. Verroust-Blondet, A., Thièvre, J., Viaud, M.: Using Euler diagrams in traditional library environments. In: Proc. ED 2004. ENTCS, pp. 189–202 (2005)
12. John, C., Fish, A., Howse, J., Taylor, J.: Exploring the notion of 'Clutter' in euler diagrams. In: Barker-Plummer, D., Cox, R., Swoboda, N. (eds.) Diagrams 2006. LNCS (LNAI), vol. 4045, pp. 267–282. Springer, Heidelberg (2006)
13. Johnson, B.: TreeViz: treemap visualization of hierarchically structured information. In: Proc. HFCS 1992, pp. 369–370. ACM Press, New York (1992)
14. Knuth, D.E.: Fundamental Algorithms, 3rd edn. The Art of Computer Programming, vol. 1, pp. 308–316. Addison-Wesley, Reading (1997)
15. Graham, M., Kennedy, J.: A survey of multiple tree visualisation. Information Visualization 9, 235–252 (2010)
16. Lamping, J., Rao, R.: Laying out and visualizing large trees using a hyperbolic space. In: Proc. UIST 1994, pp. 13–14 (1994)
17. Robertson, G., Cameron, K., Czerwinski, M., Robbins, D.: Polyarchy visualization: visualizing multiple intersecting hierarchies. In: Proc. HFCS 2002, pp. 423–430. ACM Press, New York (2002)
18. Furnas, G.W., Landauer, T.K., Gomez, L.M., Dumais, S.T.: The vocabulary problem in human-system communication. Commun. ACM 11, 964–971 (1987)

Towards the Involvement of End-Users within Model-Driven Development*

Francisca Pérez, Pedro Valderas, and Joan Fons

Centro de Investigación en Métodos de Producción de Software
Universitat Politècnica de València
Camino de Vera, s/n, 46022 Valencia, Spain
{mperez,pvalderas,jjfons}@pros.upv.es

Abstract. The models that guide the development of software systems Model-Driven Development (MDD) are usually conceived to be used by software professional developers and they are quite difficult to be understood by end-users. In this work, we propose a method that improves the involvement of end-users within MDD approaches. Furthermore, we present an example of how each step of the method is applied to involve end-users within an existing MDD approach for developing smart homes.

1 Introduction

Model Driven Development (MDD) promotes to capture every important aspect of a software system through appropriate models [1] in order to guide the full process of software development until code is obtained. MDD approaches are usually supported by Software Professionals (SPs) to specify the models from a description of end-users' needs. However, many times there are differences between the end-user description and the developed system obtained from the models. One of the reasons why this happens is that SPs may misunderstand end-user descriptions and may create models that do not represent the end-user's needs in a proper way. To improve this problem, end-users should be involved within different stages of MDD approaches.

However, involving end-users within an MDD approach is a difficult task because they do not know about Domain Specific Languages (DSLs) and modelling tools as software professionals do. Thus, the aim of our work is to bridge the gap between end-users and MDD approaches. As one of the goals of MDD is to automatically transform models into code, those models that are enriched by end-users are transformed into the final implementation. Thus, the developed application better fits the end-user expectations because end-users have the knowledge about the problem domain [2] and their involvement helps them to adopt and use the system [3].

Towards the involvement of end-users, we present a method that provides mechanisms for both end-users and SPs to allow them to work cooperatively

* This work has been developed with the support of Ministerio de Ciencia e Innovación (MICINN) under the project EVERYWARE TIN2010-18011.

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 258–263, 2011.

from the modelling stage by means of modelling and variability techniques, and specific tool support. Next, the end-user descriptions are transformed into the models of the given MDD approach. Thus, these models are enriched with end-user and SP descriptions. To sum up, the contributions of this paper are: (1) an analysis of good practices to involve end-users within MDD approaches, (2) a method to involve end-users within existing MDD approaches, and (3) an example of application of our proposed method within an existing MDD approach for developing smart home systems.

The rest of this paper is structured as follows: Section 2 briefly describes the good practices in EUD, and the six steps of the proposed method. Finally, Section 3 concludes the paper.

2 Involving the End-Users within MDD Approaches

We have identified good practices in EUD to involve end-users who have certain development skills but they only develop software to solve the specific problems that they face. These good practices are the following:

End-users should be provided with a Domain-Specific Visual Language (DSVL) [4]. The End-user Development community claims that a DSVL lowers the barriers for end-users to describe domain-specific content.

End-users should focus on user-dependent properties, whereas software engineers should focus on quality or maintenance properties [2] because end-users do not pay attention to software quality as software engineers do.

End-users have to use a library of components as a starting point in order to customize their system [5]. It is essential that a library of components or an initial system be provided by SPs.

End-users do not have to be transformed into software professionals[1]**.**

End-users have to be supported by specific tools made especially for them. Nielsen [6] recommends that end-users should participate in the description of their system though user interfaces. These interfaces should include good mappings between the user's conceptual model of the information and the computer's interface for it.

To apply these good practices, and allow SPs and end-users to work cooperatively, we have designed a method by taking as input (1) a DSL language, and (2) a MDD approach that are used by a SP. The result of applying our method (output) is (1) the DSL language and the MDD approach taken as input, (2) a DSVL that is used by end-users by means of specific tool support, and (3) modelling and variability techniques.

The method can be applied within any MDD approach where the participation of end-users should be carried out. To do this, we have proposed to perform the following steps: **Step 1)** Identify user-dependent properties to delimit the

[1] http://eusesconsortium.org/

participation of end-users in the system description; **Step 2)** Select a DSVL to allow end-users to describe the user-dependent properties; **Step 3)** Design a base system to be completed by the end-user descriptions; **Step 4)** Design reference components to provide end-users with a library to help them in the system description; **Step 5)** Define mappings between the DSL and the DSVL; and **Step 6)** Provide specific tool support for end-users.

Note that Step 1, Step 3, and Step 4 depend on the MDD project where the steps are performed. However, the remaining steps, Step 2, Step 5, and Step 6 are reused if they are performed in a different MDD project but the MDD approach and the domain are the same.

As an example, we apply our method within an existing MDD approach called Pervasive Modelling Language (PervML) [7], which is focused on the development of pervasive services in the context of smart home systems. PervML requires six models to describe services (i.e., comfort and security services), devices (i.e., alarm and presence detector devices), and the location of the services and devices (i.e., living room). These models also describe the behaviour of the services and devices. More detailed information about PervML can be found in [7]. Therefore, end-users cannot participate in the description of their smart home system because end-users lack the skills to manage the technologies that PervML uses (i.e., ASL, OCL, or UML). Next, we describe how the steps of our method are performed to involve end-users within PervML:

Step 1) Identify user-dependent properties. We have to determine the properties of the MDD project that can be defined by end-users. In the proposed example, we have chosen the comfort service of the master bedroom (parents room).

Step 2) Select a DSVL. We have to provide end-users with a DSVL to allow them to describe the user-dependent properties. The DSVL can be set by SP from the scratch or it can be reused. In this example, we select an existing DSVL named Pantagruel [8]. Pantagruel follows a sensor-controller-actuator development paradigm that improves the usability of non-SP participants according to the studies assessed in [8]. Moreover, we select Pantagruel because it offers improvements in the abstract and concrete syntax with regard to PervML. For example, Pantagruel links devices that can work as a sensor or actuator to design the events of the system. This makes Pantagruel more intuitive than PervML because PervML uses sequence diagrams.

Step 3) Design a base system. The SP designs it from previous works or designs it from scratch using the base language (DSL). The base system is marked with special model elements to indicate which user-dependent properties must be included. These special model elements are marked as variable because its description depends on end-users. We suggest that the SP uses a variability language to manage (1) the variable parts of a base system model, and (2) the components that may fit into the variable parts. To do this, we use the Common Variability Language (CVL) [9]. CVL expresses variability independently of the base modelling language in a variation model. Two concepts to express variability are the following: placement fragment and replacement fragment. A placement fragment (original) of the base model (DSL) is the fragment of the model that can be completed by so-called (alternative) replacement fragments. In our case, the SP creates placement fragments in the DSL in order to allow end-users to complete them with replacement fragments. The advantage of this step is that the base system can be reused even though the configuration of the user-dependent properties changes.

By following the example introduced above, we design the base system by using PervML. Specifically, it provides 5 services to the parents room: *ParentsRoomFireSecurity, ParentsWindow, ParentsBlind, ParentsActivation, and ParentsRoomTimeManagement*. The left side of Fig. 1 shows a partial view of the base system that we have designed and the conceptual representation of the placement fragment that is designed to support the comfort user-dependent property (represented by an empty grey square).

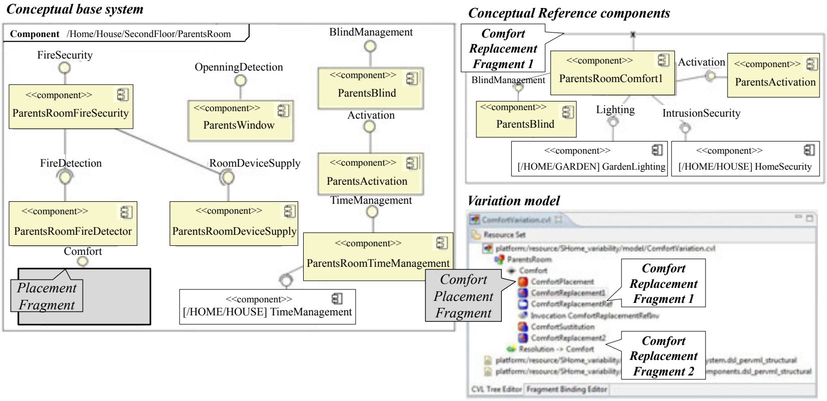


Fig. 1. Design of the comfort base system and reference components

Step 4) Design reference components. For each placement fragment that the SP has defined in the previous step, the SP can design additional reference components (replacement fragments) in order to (1) provide end-users with a library of components, or (2) use them as a starting point to customize their system. Thus, the end-users do not have to describe the system from scratch.

For the comfort fragment placement that we defined above in the variation model, we can design additional replacement fragments. For example, the functionality of a reference component is as follows: it lowers the parents' room blind, switches off the garden lights and switches on the home security when the service is activated. The conceptual view of this reference component is shown in the right side of Fig. 1. The figure also shows the variation model after the placement fragments (from the previous step) and the replacement fragments are designed.

Step 5) Define mappings. The mappings allow the SP to design the system using the DSL whereas end-users describe the user-dependent properties of the system using the DSVL. The mappings are used in a bi-directional way in order to obtain DSL descriptions from DSVL descriptions and vice versa. Thus, the

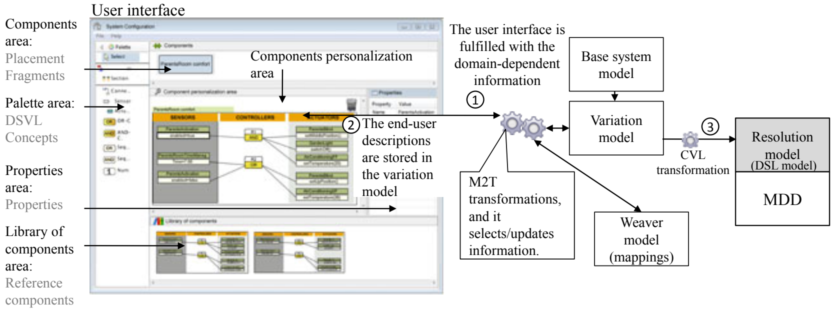

User interface
Components area:
Placement
Fragments
Palette area:
DSVL
Concepts
Properties area:
Properties
Library of components area:
Reference components
Components personalization area
The user interface is fulfilled with the domain-dependent information
1
2 The end-user descriptions are stored in the variation model
M2T transformations, and it selects/updates information.
Base system model
Variation model
Weaver model (mappings)
3
CVL transformation
Resolution model (DSL model)
MDD

Fig. 2. Specific tool support for end-users

models of the MDD approach that are taken as input at the beginning of the process are enriched by the end-user descriptions. The mappings establish relationships (i.e., links) between the DSL concepts and the DSVL concepts to store them in a model named *weaving model*. This fulfills the need to offer new custom visual languages, which according to the EUD community is essential [10].

Following the example, we set the mappings between the main concepts of PervML (*Device, Service, Interaction,* and *Trigger*) and the main concepts of Pantagruel (*Sensor, Actuator, Controller,* and *Rule*). These mappings improve both the abstract and the concrete syntax with regard to PervML. For example, two PervML concepts (*Service* and *Interaction*) that are described in different PervML models are mapped to only one Pantagruel concept (*Controller*).

Step 6) Provide specific tool support. We have developed a toolkit that provides a domain-independent user interface. This interface is fulfilled with the domain-dependent information described in the previous steps (DSVL concepts, the base system model, the variation model, and the mappings). The user interface is designed using templates that are applied from common elements, design principles, and patterns that were detected in end-user interfaces. Thus, this step provides specific tool support for end-users; and it provides elements that are domain-independent, and reusable.

Fig. 2 shows the 3 steps that the toolkit follows to support the end-user descriptions. The steps are the following:

1. It automatically fulfills the user interface to show the domain-dependent information using DSVL concepts. The left side of Fig. 2 shows the user interface and how its 5 areas (Palette, Components, Library of components, Components personalization, and Properties) are fulfilled using Pantagruel concepts.
2. It executes transformations from the user descriptions done by the DSVL concepts to DSL concepts to store the end-user description in the variation model. To do this, the toolkit uses the weaving model.
3. It makes a CVL transformation of the variation model, into a resolution model. The resolution model uses the concepts of the DSL language. Thus, the MDD process that is taken as input of the method can continue with the models enriched by end-users.

3 Conclusions and Future Work

We have presented a method that involves end-users within MDD approaches. The method follows good practices and techniques in EUD, and combines modelling techniques. We applied our method within an existing MDD approach named PervML for allowing end-users to participate in the description of their smart home. End-users can now participate in the system development by means of an appropriate DSVL and specific tool support for them. As initial evaluation, we have developed a prototype to carry out different smart home system examples that their description may involve end-users. The result is a developed application that is described using models that are enriched by both SPs and end-users. This end-user involvement since the early stages helps them to successfully adopt and use the system.

As future work, we plan to evaluate our method within the smart home domain by means of a case study with voluntary end-users that may have certain software development skills. We also plan to apply our method to others domains such as information systems, autonomic computing systems, etc.

References

1. Mellor, S.J., Clark, A.N., Futagami, T.: Guest editors' introduction: Model-driven development. IEEE Software 20(5), 14–18 (2003)
2. Costabile, M.F., Mussio, P., Parasiliti Provenza, L., Piccinno, A.: End users as unwitting software developers. In: WEUSE 2008, New York, USA, pp. 6–10 (2008), doi:10.1145/1370847.1370849
3. Rosson, M.B., Sinha, H., Bhattacharya, M., Zhao, D.: Design planning by end-user web developers. J. Vis. Lang. Comput. 19(4), 468–484 (2008), doi:10.1016/j.jvlc.2008.03.001
4. Neumann, C., Metoyer, R.A., Burnett, M.: End-user strategy programming. J. Vis. Lang. Comput. 20(1), 16–29 (2009), doi:10.1016/j.jvlc.2008.04.005
5. Segal, J.: Two principles of end-user software engineering research. In: WEUSE I Proceedings, St. Louis, Missouri, pp. 1–5. ACM, New York (2005), doi:10.1145/1083231.1083240
6. Nielsen, J.: Usability Engineering. Morgan Kaufmann Publishers Inc., San Francisco (1993)
7. Muñoz, J.: Model Driven Development of Pervasive Systems. Building a Software Factory. Phd thesis, Universidad Politécnica de Valencia (2008)
8. Drey, Z., Consel, C.: A Visual, Open-Ended Approach to Prototyping Ubiquitous Computing Applications. In: Proceedings of PERCOM 2010, Allemagne (2010), http://hal.inria.fr/inria-00484083/en/
9. Haugen, Ø., Møller-Pedersen, B., Oldevik, J., Olsen, G.K., Svendsen, A.: Adding standardized variability to domain specific languages. In: SPLC 2008 Proceedings, USA, pp. 139–148 (2008)
10. Guerra, E., de Lara, J., Díaz, P.: Visual specification of measurements and redesigns for domain specific visual languages. J. Vis. Lang. Comput. 19(3), 399–425 (2008), doi:10.1016/j.jvlc.2007.09.002

Extending the Meta-design Theory: Engaging Participants as Active Contributors in Virtual Worlds

Benjamin Koehne, David Redmiles, and Gerhard Fischer

Donald Bren School of Information and Computer Sciences,
University of California at Irvine, 6210 Donald Bren Hall
Irvine, CA 92697-3425 USA
{bkoehne,redmiles}@ics.uci.edu, gerhard@colorado.edu

Abstract. Meta-design has emerged as a theoretical framework by supporting open systems that allow end-users to become designers in dynamic use contexts. Our research is grounded in the objectives that (1) the meta-design theory can inform the design of virtual worlds, and that, in return, (2) observations and insights from virtual worlds can broaden the meta-design theory. Our work is based on 80 hours participant observation and additional interviews.

Keywords: Meta-design, virtual worlds, end-user design, human-centered design, science of design, open evolvable systems, empowerment of the user.

1 Introduction

In recent years, persistent virtual 3D-environments, in the following referred to as 'virtual worlds (VW)', have drawn increasing interest both from industry and academia [1,2,3]. VWs provide persistent virtual environments in which users can interact by controlling a virtual avatar. The avatar can move in the VW and interact with other avatars and virtual artifacts. Two of the most successful and widely used applications of VWs are:

- *massively-multiplayer online role-playing games* (MMORPGs), such as *'Lord of the Rings Online' (LOTRO)*, where millions of players worldwide come together and spend considerable amounts of time in order to engage in collaborative tasks or simply to engage in social interactions; and
- *open-ended VWs*, such as *Second Life*[1] and *OpenSimulator*[2], where users engage in many endeavors paralleling and augmenting daily life and work, including advanced, simulated environments that support various research efforts (e.g. Center for Computer Games & Virtual Worlds at UC Irvine[3]).

[1] See: www.secondlife.com

[2] See: www.opensimulator.org

[3] See: http://cgvw.ics.uci.edu

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 264–269, 2011.

In this paper we re-examine and evolve the current theories of meta-design based on observations made within VWs. We present accounts of both a gaming-oriented and an open-ended VW system. We juxtapose our observations in the MMORPG 'Lord of the Rings Online' (LOTRO) and the open-ended environment of Second Life (SL). A series of interviews with individual players of LOTRO and users of SL provide individual viewpoints and contextualize the ethnographic investigation.

2 Virtual World Systems: Second Life and LOTRO

SL represents an open-ended VW system that allows users to modify and extend the virtual environment with high flexibility and in great detail. Users build up virtual artifacts from simple building blocks and can customize appearance and behavior of these objects. SL does not prescribe a common goal for its users. Instead users explore the possibilities of the environment on their own accord.

LOTRO attracts users with a complex storyline and visually appealing graphics. Users of LOTRO can engage in various activities that ultimately lead to the development of virtual characters within the given regulations of the game framework. The creation of virtual artifacts in LOTRO is tied to the game logic and more constrained compared to open-ended VWs such as SL.

We see a dual relationship between VWs and concepts of meta-design that further motivates our argument to consider VWs for the extension of meta-design theory in general. In this discussion we specifically focus on user participation. Based on existing meta-design concepts, VWs can employ scaffolding processes that guide casual users. Similarly, the unique characteristics of popular VWs and the users they attract can inform the extension of user participation theory.

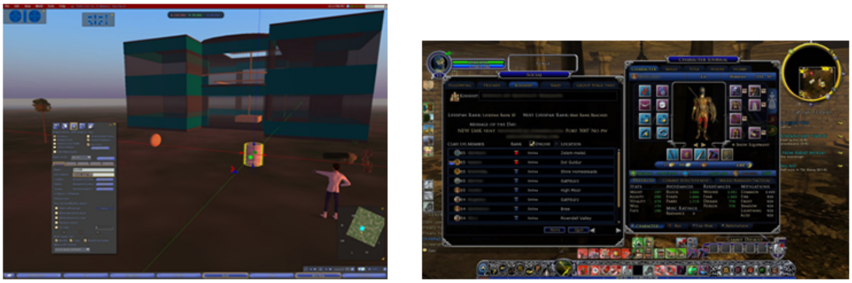


Fig. 1. Second Life (left) and LOTRO (right)

VWs have great potential to affect meta-design theory by providing environments that enable casual users to, possibly unintentionally, engage in meta-design practice.

3 Methodology

Our investigation of LOTRO and SL to re-examine and evolve the current theories of meta-design is focused on the following research objectives:

- develop *additional examples* of meta-design for worlds that have no laws and boundaries;
- support the *empowerment of end-users* that are not initially interested or motivated to conduct design practice;
- assess the *duality between VWs and meta-design*, i.e.: how does meta-design affects practices in VWs and vice versa; and,
- analyze the *support for meta-design* in both unique environments, focusing on the benefits and shortcomings of the gaming-oriented and the open-ended environment under study.

Our methodology draws from virtual ethnography [6,10] and from multi-sited ethnography [9], which allows us to supplement our findings with interviews with users outside of the VW context.

We conducted 80 hours of participant observation in LOTRO and SL. We took part in various everyday activities in both VWs with other users. Additionally, we conducted 6 semi-structured interviews with players of LOTRO and two interviews with designers in SL. Each interview lasted between 40 and 60 minutes.

The long-term participant observation allowed us to learn about individual activities of the users in the VWs. The semi-structured interviews provided us with personal perspectives and reflections. We recruited informants for the interviews through snowball-sampling in the player community of LOTRO and SL. The interviews focused on themes of the VWs that we related to meta-design activities based on our previous participant observation. Additionally, we asked the participants about their individual personal backgrounds and their general motivations to participate in the VWs.

Patterns and themes based on our observations were analyzed using open-coding techniques [8]. The meta-design concepts identified in the following chapter were used to filter and code the collected data. The data analysis focused on practices framed by meta-design in the VWs. We were specifically interested in the changing roles during different stages of participation. We structure our findings based on central concepts of meta-design originally formulated in [5].

4 Discussion: Meta-design Concepts in SL and LOTRO

SL and LOTRO represent two systems that implement elements of meta-design in quite different ways. Both systems can individually serve as systems for exemplifying various concepts of meta-design theory. A synthesis of the exemplified concepts allows for a more detailed analysis of VW system features that particularly affect casual end-user participation. Table 1 provides a comparison of the central meta-design concepts and the corresponding properties of the VW systems included in the analysis. An expanded discussion of these concepts is available in [7].

Table 1. Meta-Design Concepts in Virtual Worlds

Meta-Design Concept	LOTRO	Second Life
convivial tools	Adventure & leveling system	prim design tools
domain orientation	goals from fixed set in the fantasy world context	user-imagined goals, open-endedness
open, evolvable systems	limited customization	unrestricted customization
underdesigned systems	fixed fantasy world context	minimalist environment
collaborative work practices	high cooperation amongst players	limited cooperation, specialization

VWs, in particular MMORPGs, draw a large number of users. As our analysis has shown, users join the environments with very different aspirations. VWs can accommodate a very diverse user base by providing a system of technological and social structures that is sufficiently flexible to provide individuals with discrete virtual spaces within the larger system. Current systems discussed in the context of meta-design mostly lack opportunities to unpack rich ecologies of participation. Our observations in the VWs point to a collaboration practices as deciding factors to support different levels of participation in the game- and open-ended context.

LOTRO makes casual gamers gradually aware of the functional and social properties of the game. The scaffolding system in LOTRO is based on elements in the VW providing guidance and works in close connection with collaboration between the players. While SL represents an open environment with great freedom of creative interaction, LOTRO's strength lies in the integration of collaborative community effects.

VWs offer an opportunity to study the effects of collaboration on the way casual users move through ecologies of participation. Technical scaffolding systems alone are not sufficient. Instead, social community components need to make collaboration tools more accessible and attractive for casual users.

Collaboration in VWs is closely tied to the social structure of the system. In LOTRO, collaboration is initially triggered by the functional structure of the game that creates the need for players to find collaborators for difficult adventures or to create powerful artifacts. Over time, the recurring need for collaborative activities creates social networks amongst players that, once initially established, are sustained for a longer time period. In this eco-system of social collaboration, the motivation to help others becomes detached from traditional reward systems. Respect and a good reputation become equally or more important than monetary rewards or other character improvements.

These examples show that a useful extension of the meta-design framework would be a careful analysis of user roles on an individual level. Intrinsic motivations to engage in creative activities only come forward in large scale systems such as LOTRO or SL.

Users of VWs can dynamically switch between the roles of passive consumers and active contributors. While the degree of contributions differs between LOTRO and SL, it is common in both worlds that users are not locked into either the designer or the user role.

The domain-oriented gaming environment creates opportunities for need-based attendance to creative practices. Players can also decide to completely neglect crafting activities. The open-ended environment of SL allows users to switch between observer and designer roles. However, the complexity imposed by the available design tools creates a higher entry barrier to become a designer. Meta-designers need to find a middle ground between complex and universal design tools as exemplified in SL and socially embedded creative activity as exemplified in LOTRO. Our results point to a priority for socio-technical contexts that allow users to develop their roles based on social interactions through collaborative activities with other users. More experienced users develop their own strategies while novice users require guidance to be gradually introduced to the system functionalities. There is not a perfect solution for this trade-off problem.

However, our analysis of two distinct VWs leads us to suggest an extension of the seeding, evolutionary growth and reseeding model (SER) [4] that clearly focuses on support during transition phases between different stages of participation. In the VWs other users constantly provide seeds for design practice. The leveling system in LOTRO gradually prepares players with coordinated stages and sub-goals on their progression from novice players to experienced performers. Combined with the visibility of fine-grained design processes observed in SL, our findings suggest a combination of these two concepts. Designers and design artifacts undergo stages of evolutionary growth. Design products are re-introduced in the environments and serve users on different levels of participation as examples for novel design projects.

Meta-design theory can benefit designers of VWs. The meta-design concepts discussed in the qualitative investigation can guide game designers but also designers of open-ended systems. Possible design recommendations can be imagined by combining the social community elements found in LOTRO and the accessible design tools in SL. Studying VWs can contribute to broadening the theory of meta-design. VWs inherit qualities that can inform an expansion of the theory by providing additional systems to exemplify meta-design concepts. The large numbers of casual users with diverse interests in our view represent the missing masses that meta-designers should focus on. Gaming-oriented VWs employ concepts that gradually empower end-users while keeping them motivated and engaged. Social structures create an environment that fosters cooperation amongst the players.

Meta-designers look for tools to empower end-users to tailor systems towards their needs. Open-ended environments like SL provide these types of tools in virtual spaces. The tools do not discriminate between novice users and users with design experience in different domains. Studying tools that create an even, flat entry level for all users can inform concepts of meta-design tools in general.

Based on our study, we have identified a rich duality between VWs and meta-design. Meta-design potentially provides VW designers with a useful analytic design for user participation and open systems. VWs extend meta-design theory by providing a class of new systems to exemplify meta-design concepts.

Initially, it appears simple to dismiss LOTRO as a system for meta-design practices based on the regulations imposed on the users and the limited design opportunities in the VW. However, our analysis has brought to light concepts that meta-designers are well-advised to take into account.

5 Conclusion and Future Work

This work opens many avenues for future research. First, some open-ended VWs such as SL and, more particularly, the open source counterpart, OpenSimulator, provide means for extension through source code modification. Extending them to provide scaffolding that makes meta-design more explicit in the VWs that players or end users experience is one real possibility. Second, the potential that meta-design holds for empowering end users is an on-going exploration. The implications in altering the power between the roles of end users and designers are just being understood.

Acknowledgments. This work was supported by the U.S. National Science Foundation under grant #0808783. We would like to thank our colleagues at UC Irvine and the University of Colorado at Boulder for their insightful comments.

References

1. Bainbridge, W.S.: The Scientific Research Potential of Virtual Worlds. Science 317(5837), 472–476 (2007)
2. Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A.: A meta-design approach to end-user development. Paper Presented at the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (2005)
3. Fischer, G.: End-User Development and Meta-design: Foundations for Cultures of Participation. In: End-User Development, pp. 3–14 (2009)
4. Fischer, G., Grudin, J., McCall, R., Ostwald, J., Redmiles, D., Reeves, B., Shipman, F.: Seeding, evolutionary growth and reseeding: The incremental development of collaborative design environments. In: Olson, G.M., Malone, T.W., Smith, J.B. (eds.) Coordination Theory and Collaboration Technology, pp. 447–472. Lawrence Erlbaum, Mahwah (2001)
5. Fischer, G., Scharff, E.: Meta-design: design for designers. In: Proceedings of the 3rd Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques. ACM, New York (2000)
6. Hine, C.: Virtual ethnography. SAGE, London (2000)
7. Koehne, B., Redmiles, D., Fischer, G.: Details on Extending the Meta-Design Theory: Results from Participant Observation of Active Contributors in Virtual Worlds (2011), `http://www.isr.uci.edu/tech_reports/UCI-ISR-11-1.pdf` (March 14, 2011)
8. Lofland, J.: Analyzing social settings: a guide to qualitative observation and analysis, 4th edn. Wadsworth/Thomson Learning, Belmont, CA (2006)
9. Marcus, G.E.: Ethnography in/of the World System: The Emergence of Multi-Sited Ethnography. Annual Review of Anthropology 24, 95–117 (1995)
10. Nardi, B.A.: My life as a night elf priest: an anthropological account of world of warcraft. The University of Michigan Press, Ann Arbor (2010)

Community Network 2.0: Visions, Participation, and Engagement in New Information Infrastructures

John M. Carroll, Michael Horning, Blaine Hoffman, Craig Ganoe,
Harold Robinson, and Mary Beth Rosson

Center for Human-Computer Interaction
The Pennsylvania State University
University Park, Pennsylvania 16803 USA
jmcarroll@psu.edu

Abstract. Through the past seven years, our research group has engaged in a participatory action research collaboration with a variety of community partners to explore understandings, possibilities, and commitments for a new community networking infrastructure in State College, Pennsylvania. This paper describes a case study of multifaceted information technology infrastructures, and of collaborating with the plethora of actors and institutions that are stakeholders in such infrastructures. Information technology projects increasingly depend upon the commitment and energies of a great diversity of stakeholders. Understanding better how broad projects move forward is critical to society.

Keywords: Participatory Action Research, Information Infrastructures, Community Networks, Location-based Services, Feed Aggregation.

1 Introduction

Community networks provide online tools and information for users living in proximity. Early networks used simple text tools like bulletin boards or email to invite opinions and concerns [1]. In the 1990s, community networks migrated to the web, with a brief period of growth. They exploited the accessibility of the Internet, and the expressive power of HTML. There were ironies in this; for example, posting information became easier, but community discussion became less easy. But the internet quickly attracted commercial interests on a global scale. Soon, websites of local merchants and nonprofits pointed to corporate portals. Government information migrated to government sites. Tourism information migrated to tourism sites. By the early 2000s, the concept of community network had fragmented into a chaos of redundant and commercial or semi-commercial portals, often carelessly maintained.

The overarching objectives of community networking – enhancing participation of end users in designing their community technology; exchange of community information - remain valid today. However, user expectations about web information systems are much more demanding. People expect up-to-date information, value-added interactions and new options like wireless access, location-based services, syndication,

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 270–275, 2011.

recommendation and peer sharing. Support of these richer interactions and information requires much more than HTML editors. Our research team has been exploring options such support for the past seven years, working with a wide variety of community partners in State College, Pennsylvania.

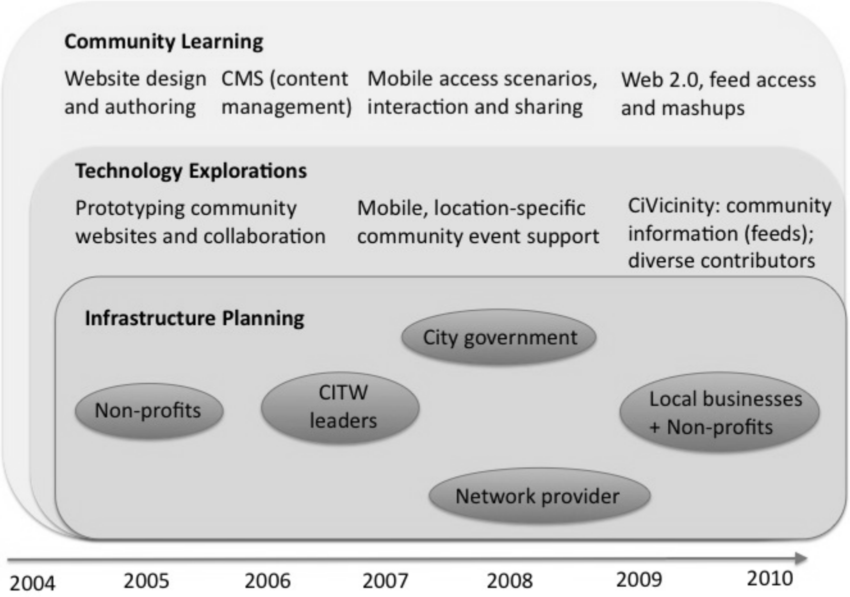

Community Learning
Website design and authoring
CMS (content management)
Mobile access scenarios, interaction and sharing
Web 2.0, feed access and mashups
Technology Explorations
Prototyping community websites and collaboration
Mobile, location-specific community event support
CiVicinity: community information (feeds); diverse contributors
Infrastructure Planning
City government
Non-profits
CITW leaders
Local businesses + Non-profits
Network provider
2004
2005
2006
2007
2008
2009
2010

Fig. 1. Three threads in our project, showing relative timing and embedded relationships

In this short paper, we summarize our participatory activities by narrating three overlapping threads Fig. 1: (1) A community learning process in which community members reflected on technology possibilities and developed skills; (2) Our own technology explorations of specific services, such as location-based wireless and feed aggregation; and (3) Infrastructure planning with community leaders to help them envision possibilities and become stakeholders in the implementation process.

2 Community Learning

When we began this project in 2003, many of our local community groups were using websites to increase their visibility, or to organize key functions like announcements and donations. We began to explore how community groups were articulating technology goals, learning IT skills, and building practices with respect to novel information technology. We structured our investigations into cycles of 2-year partnerships, recruiting four local nonprofits for each cycle. The groups learned by reflecting on their current use of the Web, and specifying information technology tasks they wanted to accomplish. They also analyzed the organizational structures they would need for addressing such challenges in a continuing way. In return, we facilitated their planning processes and pursuits of specific technology needs.

The outcomes of our community learning investigations varied through the years. Early on, we documented fascinating examples of how even technologically sophisticated groups were often disempowered with respect to their own information technology. One community development group that focuses on water quality issues had hired a local web designer. The contractor designed a website that the group considered a cliché and not an expression of their mission [2]. They refused to even allow the site to go public because of the visual design had bamboo in its background – "an invasive species […] stuff we pull out". The contracting also left them powerless; the contractor hosted their content on his server, and was not responsive to their concerns.

Some challenges were management issues, for example having no technology plan at all. Their reasoning was that it did not make sense to plan for technology when they had so few resources to invest in it. Many groups relied on volunteers as their webmasters and system administrators, but volunteers tend to come and go in nonprofit groups. Other challenges were identity issues: One group leader told us flatly that no one joins a local nonprofit group to manipulate software [3].

The informal learning engagements had both short- and long-term implications. Group members became more comfortable, literate, and skilled with respect to web technologies like HTML and low-tech design methods like sketching and scenarios. The community development group was able to use such methods to envision and build a new and more appropriate website. In the longer term, groups felt greater autonomy and control of their own web information technology, and began to develop practices that could sustain transformation. The community development group subsequently decided to build technology-related knowledge management practices within the group itself, reducing its future dependence on outside technical experts.

In addition to the engagements with nonprofits, we initiated a series of community information technology workshops (CITWs), to recruit partners and disseminate findings to the larger community. At these events, we and other community members provided tutorials and demonstrations of new technologies and approaches. There have been five such workshops: October 2003, October 2004, August 2005, August 2006, and April 2010. The first four CITWs progressed from fairly small and focused on recruiting partner organizations and sharing results to broad discussions of the community nonprofit organization's information technology needs and resources and half-day and full-day tutorials on new technologies (see [4] for a more complete description and analysis of CITWs 1-4).

The most recent workshop, CITW 5, introduced Web 2.0 technologies to the community, for instance as an aid to collecting and organizing group information or enabling more effective outreach to community residents. For instance, we demonstrated how Google's online documents and calendars can be recruited as lightweight collaboration and communication tools. Tools for creating and using RSS and calendar feeds were demonstrated as a way to publicize issues and events. While these Web 2.0 services are used many web-savvy digital natives, the idea of using such services for community outreach goals was a novel concept.

To sum up, we learned that nonprofits face significant challenges with respect to IT. They have few IT resources because they have few resources in general. They have no IT management practices, and often lack specific technical skills. However with modest support, our partners were eager and creative about adapting their IT practices, particularly web-related IT innovations. Through the CITW process we observed that nonprofits in general are quite receptive to community discussions about technology skills and practices, unmet challenges, and new possibilities.

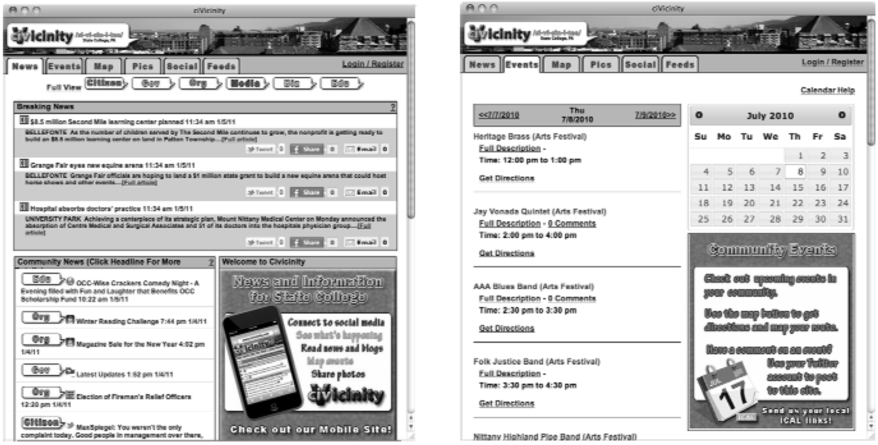
3 Technology Explorations

Through a series of empirical requirements studies and prototypes we are currently exploring how community members might appropriate new technologies (e.g., aggregation of local information or location-based wireless services). We began with a series of interviews, surveys, and focus group discussions with community groups [5]. We observed a high level of interest in our design scenarios, but also apprehension about the perceived difficulty of managing mobile interactions. We concluded that we needed to involve people in more concrete experiencs of technological possibilities.

To help community members grasp how technology can support community activities, we built prototypes providing location-sensitive, mobile information for community events [6]. Our first was a wiki-style mobile system with an interactive calendar of events. We conducted field trials of the prototype during two popular community events: the 2008 Central PA Festival of the Arts in the summer and First Night State College 2009. We recruited a small number of residents to take mobile devices to these events so that they could try out the prototype and provide feedback.

These trials were a step in the right direction, but we were entering and managing the information by hand and thus were responsible for keeping it current. For a time we partnered with a commercial provider, StateCollege.com, which already hosted and maintained a community calendar. While functionally similar to our first prototype application, the new tools enabled users to submit events that are merged into the database. Once in the database, these events appear on the calendar, and are integrated within the StateCollege.com site; we thus inherited mobile and desktop views, user commenting, and search. In short, we merged our prototype's features with theirs to better achieve our aim as well as improving their community calendar. This prototype was field tested during the 2009 Central PA Festival of the Arts. Unfortunately, this partnership eventually floundered: In retrospect, we recognize that while StateCollege.com calendar service is *for* the community, it is not *by and of* the community.

So, we took location-sensitive content and community ownership of information as primary design points and built the CiVicinity prototype (Fig. 2). Our new design relies on Web 2.0 services that enable direct participatory development of the community information system. Community groups use RSS or ATOM to publish information that is aggregated by CiVicinity. We enhance the content they publish with location-specific features that apply not only to events but also any place-related community content. Residents "program" custom views of the resulting content by assembling just the feeds (or categories of feeds) they prefer. Our primary goal is to support collaboration for citizens to develop, share and maintain information as part of their own community system. The screenshot shows the home page as well as a more focused event view. CiVicinity is in operation but continuing to evolve and expand as we interact with stakeholders and gather feedback.



Breaking News

$8.5 million Second Mile learning center planned 11:34 am 1/5/11

BELLEFONTE As the number of children served by The Second Mile continues to grow, the nonprofit is getting ready to build an $8.5 million learning center on land in Patton Township....[Full article]

Grange Fair eyes new equine arena 11:34 am 1/5/11

BELLEFONTE Grange Fair officials are hoping to land a $1 million state grant to build a new equine arena that could host horse shows and other events....[Full article]

Hospital absorbs doctors' practice 11:34 am 1/5/11

UNIVERSITY PARK Achieving a centerpiece of its strategic plan, Mount Nittany Medical Center on Monday announced the absorption of Centre Medical and Surgical Associates and 51 of its doctors into the hospitals physician group....[Full article]

Community News (Click Headline For More

Edu OCC-Wise Crackers Comedy Night - A Evening filled with Fun and Laughter that Benefits OCC Scholarship Fund 10:22 am 1/5/11

Org Winter Reading Challenge 7:44 pm 1/4/11

Org Magazine Sale for the New Year 4:02 pm 1/4/11

Gov Latest Updates 1:52 pm 1/4/11

Org Election of Fireman's Relief Officers 12:20 pm 1/4/11

Citizen MaxSpiegel: You weren't the only complaint today. Good people in management over there,

Welcome to Civicinity

Calendar Help

<<7/7/2010 Thu 7/8/2010 7/9/2010>>

Heritage Brass (Arts Festival)
Full Description -
Time: 12:00 pm to 1:00 pm
Get Directions

Jay Vonada Quintet (Arts Festival)
Full Description - 0 Comments
Time: 2:00 pm to 4:00 pm
Get Directions

AAA Blues Band (Arts Festival)
Full Description - 0 Comments
Time: 2:30 pm to 3:30 pm
Get Directions

Folk Justice Band (Arts Festival)
Full Description -
Time: 3:30 pm to 4:30 pm
Get Directions

Nittany Highland Pipe Band (Arts Festival)

July 2010

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Community Events
Check out upcoming events in your community.
Use the map button to get directions and map your route.
Have a comment on an event?
Use your Twitter account to post to this site.
17
Send us your local ICAL links!

Fig. 2. CiVicinity news feed page (left) and Arts Festival calendar (right); http://civicinity.org

4 Infrastructure Planning

After CITW 3, a group of community leaders initiated a planning process directed at creating a more continuous community learning mechanism and a more comprehensive information infrastructure for nonprofits, and for the community in general. These efforts were critical in organizing and realizing subsequent CITWs. They also helped to focus community discussion about shared technology infrastructures [4].

After CITW 5, a group of partners that included the regional public library, local public access media, and a youth services bureau began meeting to plan a comprehensive community calendar. They wanted software that could support a wide variety of information types in order to maximize adoption from local nonprofits. This melded well with our ongoing prototyping of CiVicinity and provided additional motivation for supporting a broad range of Web 2.0 APIs. The meetings also addressed the need for a wireless network that could enable community information input and sharing. We faced the challenge of building such a network without town council support – an earlier plan to implement a wireless network had recently been abandoned. We agreed that demonstrating the *value* of community wireless access was of high priority.

Of immediate concern was showing that a downtown wireless network could provide opportunities for a wide range nonprofits, businesses and government entities to interact with community members and visitors. This led the group to emphasize geo-location services. Such services can enhance civic, social and economic interests and support many new kinds of interactions on mobile computing platforms, including location-based advertising benefits local businesses. Our community partners are not commercially motivated, but they see that a collective venture between local business and nonprofits may be an important step to take.

A related topic is long-term sustainability of a community wireless network. With respect to this issue, our partners expressed two areas of concern. First was technical support for hosting the network. Our community partners who have IT expertise are already spread quite thin in their organizations, while also volunteering additional services for other community groups. Thus while eager to support network development, they worried about long-term support. Our partners also emphasized the need for a business model that can support both implementation and maintenance of the network. We are currently facilitating an cooperation between the community groups and an organization of downtown merchants to address these issues.

5 Summary and Concluding Remarks

The internet has dramatically expanded the possibilities for information systems. Recent developments like Web 2.0 services enable enhanced participation and end-user design. In this paper we summarized a participatory action research project with nonprofit groups in State College, Pennsylvania, more recently including representatives of local businesses. We presented the project through three related threads - community learning, technology explorations, and infrastructure planning.

Each thread of participatory development has led to recruitment of important community allies. In many cases, the three threads have been symbiotic: The community learning produced more ambitious goals. Our technology explorations helped our partners to grasp new possibilities and realize they have key roles to play in fulfilling these possibilities. Our partners' evolving technology goals have helped community planners recognize the need for an alliance among nonprofits and businesses. In the end, technology can entrain cultures of participation only when it is integrated into multi-faceted social and institutional infrastructures, predicated upon the interdependence of technology and human activity, and aimed at enhancing participation and engagement among the greatest possible variety of human actors.

References

1. Schuler, D.: New community networks: Wired for change. Addison-Wesley, New York (1996)
2. Farooq, U., Ganoe, C.H., Xiao, L., Merkel, C., Rosson, M.B., Carroll, J.M.: Supporting community-based learning: Case study of a geographical community organization designing their web site. Behaviour & Information 26(1), 5–21
3. Merkel, C., Farooq, U., Xiao, L., Ganoe, C., Rosson, M.B., Carroll, J.M.: Managing technology use and learning in nonprofit community organizations: methodological challenges and opportunities. In: Proceedings of CHIMIT 2007 (2007)
4. Carroll, J.M., Bach, P., Rosson, M.B., Merkel, C.B., Farooq, U., Xiao, L.: Community IT Workshops as a Strategy for Community Learning. First Monday 13(4) (2008)
5. Burge, J.D., Campbell, L.M., Carroll, J.M.: Community wireless networks: Field exploration of non-profit participation. In: HCI Symposium at ASIST 2009, October 26, Columbus, Ohio (2009)
6. Carroll, J.M., Ganoe, C.H.: Supporting Community With Location-Sensitive Mobile Applications. In: Handbook of Research on Urban Informatics: The Practice and Promise of the Real-Time City. Information Science Reference, IGI Global, Hershey, PA (2008)

Meta-design Blueprints: Principles and Guidelines for Co-design in Virtual Environments

David Díez, Paloma Díaz, and Ignacio Aedo

DEI Lab - Computer Science Department
Universidad Carlos III de Madrid, Spain
{ddiez,pdp}@inf.uc3m.es, aedo@ia.uc3m.es

Abstract. Meta-design is defined as a conceptual framework that allows end-users to create contents by using socio-technological infrastructures in which people can actively participate. Despite the fact that the foundations for meta-design are well known, it is still an abstract concept lacking of suitable design artifacts to guide its application. The aim of this paper is to analyze the concept of meta-design in order to define blueprints for its application in web-based environments. Based on a review of both literature and websites conceived to design contents by users, the paper compiles a number of guidelines to address the development of web tools that support meta-design.

Keywords: meta-design, designing design, guidelines, web-based environment.

1 Introduction

One of the main weaknesses of designing is related to the own evolutionary character of the design and the incapacity of fully anticipating the needs and tasks of users at design-time [5]. With the purpose of overcoming this limitation, meta-design aims at defining mechanisms that allow 'owners of the problem' to become designers [1]. Meta-design is defined as a '*conceptual framework aimed at defining and creating social and technical infrastructures in which new forms of collaborative design can take place*' [3]. Meta-design is therefore oriented to fulfill not only technological artifacts but also social contexts that allow users to participate in the process of design [2,4,6]. Nevertheless, though meta-design has been successfully applied in different domains, there are neither guidelines nor detailed rules that lead the creation of environments that support meta-design; being based their definition on the experience of meta-designers.

The purpose of this paper is to define a set of directions that address and systematize the creation of web-based platforms that support meta-design. The elaboration of these guidelines has relied on the review of web tools for co-creation. First of all, taking into account related literature, we defined a number of criteria for discriminating meta-design environments. Secondly, different web tools based on participation were identified, filtering them in keeping with the previous defined criteria. Once appropriate web tools were discriminated, their services were studied, identifying a set of common characteristics that support the meta-design process. Finally, these characteristics were normalized and categorized by activities, defining a set of

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 276–281, 2011.

guidelines. The rest of the paper is organized as follows. The review of web tools for co-creation is described in next section. Section three is focused on explaining the guidelines for designing the design process. Finally, conclusions and recommendations for further work are drawn in the last section.

2 A Qualitative Survey of Co-creation

The growing signification of cultures of participation and the appearance of new technologies that support this tendency in the web context have led to a profusion of web applications that enable users to actively participate in personally meaningful problems. All those web applications may be considered as a reference to identify what requirements, features, or services are required by web-based environments that support meta-design. With the purpose of achieving this goal, the review of several web tools for participation has been carried out.

The review started by identifying web tools that provide environments for the creation of contents, the publication of ideas, or the submission of edits. After a preliminary search, twenty-four tools were selected. The majority of them were websites conceived to connect people with similar passions and interests. In a second step, all those sites that support the submission of contents but not their revision, or whose rules of engagement are neither well known nor explicit, were withdrawn; similarly, since there were several websites to share hobbies, we decided to filter them and just consider one of the most popular websites –'Do It Yourself Happy'- about pastimes. In addition, because of funding restrictions to carry out the study, not free web sites were removed. At the end of the selection process, eleven representative web tools had been selected.

The following step was based on analyzing the capabilities of those tools to support the evolution of solutions. As a result, we decided to remove four additional websites that provide mechanism to create contents and share knowledge among distributed participants, but not allowing the evolution and gradually improvement of the solution. Finally, we went over the other seven[1] web sites, analyzing their capabilities, features, and services. Services such as comment, annotation, tagging, and rating support the participation of the community and the improvement of the solution; similarly, feedback and reviewing services have proved as effective mechanisms for addressing the development. Other services, as tracking and version control, are used to manage the evolution of the solution, supporting its validation and the reverse of undesirable changes. Finally, participation rules and membership procedures are usually published as a way of resolving conflicts and clarifying behaviors. In summary, co-creation in virtual environments is based on services that allow participants to exchange ideas in a structured and organized way, promoting participation, and avoiding an uncontrolled evolution of the creation.

3 Guidelines for Building Meta-design Web Environments

From the review of the selected websites we have compiled a set of characteristics, shared by all these applications, which support the collaborative creation of contents

[1] OpenStreetMap, ABtests.com, Google Sketch Up, IkiMap, Knol, PassTheBall, Wikipedia.

and knowledge in virtual environments. These characteristics may be classified into four activities according to the objective to achieve: the elaboration of the solution, the participation in the design process, the collaboration with others by expressing yourself, and the validation of the evolution of the solution. Relying on these characteristics, it is possible to define general rules or guides that serve as a compilation of reviewed web applications. Such guidelines may lead the building of virtual environments for meta-design. This section is devoted to describe these guidelines for which they will be organized into four groups dealing with the aforementioned activities: elaborate, participate, express yourself, and validate.

- **Elaborate.** Meta-design aims at creating original solutions to significant problems. Thus, a virtual meta-design environment must provide workspaces to develop, derive or specify new products. Besides, these workspaces should provide services not only to create new contents but also to alter, edit or modify them. The design directions considered in this category are:

  - G1. Define a collaborative creation environment to produce and evolve virtual artifacts. As collaborative creation environments we may identify workspaces that allow participants to upload files, to create online documents, to produce specific resources –such as maps, surveys, data tables, presentations, etc.-, to develop single components, or to customize composite elements.

  - G2. Address the evolution of the solution. Design is defined as an evolutionary process aimed at gradually developing a solution from early models to final products. Although evolution is an essential element of meta-design, and one of the reasons of its success, designs do not freely evolve but change controlled by the community itself. As a consequence, rating mechanisms that allow member of the community to appraise contributions, point out the most relevant, and get rid of unworthy directions are required. In addition, reviewing and feedback mechanism are effective ways of controlling the evolution of the solution.

  - G3. Tracking changes. Due to its evolutionary character, participants can make continuous modifications and suggestions throughout the design process. Thus, with the purpose of managing the evolution of the product, tracking changes mechanisms are required. Such mechanisms provide a systematic record of modifications and comments related to them. Tracking include services such as logging, control versions, related changes, changes history, undoing and redoing changes, etc.

- **Participate.** Meta-design relies on participation, on exchanging and sharing ideas as a manner of meeting needs and achieving objectives. As a consequence, participation is an essential activity that should be encouraged and rewarded. Nevertheless, taking part affects and is affected by the others. Different participants have different ideas about similar situations; then, a virtual meta-design environment must provide mechanisms to deal with conflicts and misinterpretations. The guidelines in this category include:

  - G4. Award participation. Participants' contributions should be rewarded in order to recognize their efforts and achievements, and encourage further participations. There is a neither unique nor definitive way of rewarding participation; however, using ego as a grant is a very common way of rewarding. For instance, 'Knol' presents the name of the most prolific participants in the main page, as well as the list of the most popular contributions; on the other hand, 'Google Sketch Up' provides a gallery that compiles the best models according to the opinion of users.

  - G5. Ease involvement. Participation is a costly activity that should be simplified by providing multiple channels to communicate and to contribute. As an example, 'OpenStreetMap' allows participants to contribute in different ways: editing elements of maps, adding points of interest such as petrol stations, hospitals, or drugstores, and importing information about either roads or geographical points gathered by using GPS; similarly, 'Knol' provides features to upload papers, edit them, or just reviewing contributions of others. The more ways of contributions are provided by the environment the more suitable solution will be achieved by the community.
  - G6. Policies and guidelines. It is essential to define participation rules that determine in an unambiguous way how members of the community can participate and express themselves. These rules might avoid confusions as well as inappropriate contributions. Policies and guidelines should be public and accessible, and they must be accepted by the members of the community before contributing.
  - G7. Dispute resolution. Along the design process, emerging disputes among participants is almost inevitable; specially, related to the validation activity. With the purpose of avoiding unpleasant situations and guaranteeing the good atmosphere of the community, resolution guidelines and application processes must be defined. The main purpose of the community should be avoiding disputes. Meta-design is built upon the principle of collaboration and assuming that the efforts of others are in good faith is important to any community. Just in case this purpose fails, formal process or third-party intervention mechanism must be applied. Techniques such as negotiation, mediation, arbitration, and even sanction may be applied to resolve disputes. As policies and guidelines, the dispute resolution process should be published in an accessible place of the environment.

- **Express yourself.** Being member of the community requires not only participate but also express your opinions and comments as a way of influencing in others. Nevertheless, members of the community are not homogeneous. Different members have different knowledge, different expertise, or different level of implication. Then, a virtual meta-design environment must support different ways of expressing. The guidelines in this category include:

  - G8. Levels of users. All opinions are valuable but not all opinions have the same value. Moreover, depending on your background, your opinion may be expressed in a different way and should be considered in a different manner. As a consequence, the virtual environment must provide different levels of user (roles) as stated in the user profile. The definition of different levels of users concerns not only to give opinions but also to create contents or modify the contributions of others.

  - G9. Provide different ways of expressing. Give your point of view is a critical and difficult activity that should be eased by supporting different options, including annotation, tagging, rating, reviewing, etc. Similarly, managing and accessing opinions of others should be facilitated by using different mechanisms such as content filtering, ordering mechanisms, search services, etc.

- **Validate.** As aforementioned, neither the design process can freely evolve nor all contributions are fitting or applicable. For this reason, it is necessary to provide validation mechanisms that lead the evolution of the design. Validation activity aims at checking and proving the accuracy of contributions (creations, opinions, changes, etc.) in order to guarantee the achievement of common goals and the success of a better solution. Based on our review, three different ways of validation are usually applied:

  - G10. Validation by the community. It is the most important and most usual validation mechanism. Meta-design relies on collaboration and membership, on groups of people who share their creations, ideas, and opinions in order to reach common goals. Thanks to this way of working, contributions are therefore validated and self-regulated inside of the community, achieving a steady improvement.

  - G11. Validation by holders. Sometimes, depending on the product to meta-design, the solution has an owner or stakeholder who is especially interested in the direction of the evolution of the product. As examples of this situation, we can highlight 'Knol' and the authors of the original paper draft, or 'Google Sketch Up' and the designers of the original 3D model shared to other members of the community. These owners and stakeholders are responsible of addressing the evolution of their products, rejecting inappropriate changes, and prioritizing opinions, but always with an open-minded and receptive attitude.

  - G12. Validation by curators. When the validation by the community and holders is not enough, third-party validators must be involved. These validators, known as curator or supervisors, are experts or specialties appointed by the community to review the evolution of the design, reject inappropriate progressions, and revert unfitting changes. In addition, curators may behave as mediators or third-part negotiator in case of disputes among members of the community.

4 Conclusions and Further Works

The review of web tools for social creativity and mass collaboration has allowed us to identify a set of services, features and capabilities that supports the basic of meta-design: the evolutionary creation of solutions by end-users. In particular, services such as annotation, tagging and rating are essential to achieve a better solution; similarly, reviewing and feedback mechanism are effective ways of addressing the evolution of the solution. At the same time, the definition of policies and guidelines to describe best practices, clarify behaviors, and resolve conflicts are necessary to create a fruitful and prolific community of participants. Furthermore, such a community should not be considered as a homogenous reality but as a blending of different groups of participants with diverse profiles, interests, and experiences. The environment must therefore support different roles and levels of participation. On the other hand, solutions cannot freely evolve; then, it is necessary to provide mechanisms to manage the development of the solution and to assure its suitableness. Among them, tracking and version-control services are successful instruments to control the evolution of the solution, allowing undoing actions and recovering operations. Finally, participants, creators, and curators or reviewers should be able to carry out the validation of the solution according to well-known procedures and rules.

Further work will lead to the development of technological platforms that allow the evaluation of our guidelines for building meta-design environments, as well as their refinement or refutation. The final aim of our work is to define a design framework that addresses the application of the meta-design paradigm in the web context.

Acknowledgements

This work has been partly supported by the urTHEY project (TIN2009-09687) funded by the Ministry of Science and Innovation (MICINN) of the Government of Spain.

References

1. Costabile, M.F., Fogli, D., Lanzilotti, R., Marcante, A., Mussio, P., Parasiliti Provenza, L., Piccinno, A.: Meta-design to face co-evolution and communication gaps between users and designers. In: Stephanidis, C. (ed.) HCI 2007. LNCS, vol. 4554, pp. 46–55. Springer, Heidelberg (2007)
2. De Kerckhove, D.: Networked Art and Virtual Communities. In: Faure, C. (ed.) Arslab: I Sensi del Virtuale, pp. 103–110 (1995)
3. Fischer, G.: Distances and Diversity: Sources for Social Creativity. In: Proceedings of Creativity & Cognition, pp. 128–136. ACM Press, New York (2005)
4. Fischer, G.: End-User Development and Meta-Design: Foundations for Cultures of Participation. Journal of Organizational and End User Computing 22(1), 52–82 (2010)
5. Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A.G., Mehandjiev, N.: Meta-Design: A Manifesto for End-User Development. Communications of the ACM 47(9), 33–37 (2004)
6. Youngblood, G.: Metadesign: Toward a Postmodernism of Reconstruction. Ars Electronica Catalog. Linzer Veranstaltungsgesellschaft (1986)

End-Users Productivity in Model-Based Spreadsheets: An Empirical Study*

Laura Beckwith[1], Jácome Cunha[2,3], João Paulo Fernandes[2,4], and João Saraiva[2]

[1] HCIResearcher, Denmark
beckwith@hciResearcher.com
[2] Universidade do Minho, Portugal
{jacome,jpaulo,jas}@di.uminho.pt
[3] ESTGF, Instituto Politécnico do Porto, Portugal
[4] Universidade do Porto, Portugal

Abstract. Spreadsheets are widely used and studies show that most of the existing ones contain non-trivial errors. To improve end-users productivity, recent research proposes the use of a model-driven engineering approach to spreadsheets.

In this paper we conduct the first empirical study to assess the effectiveness and efficiency of this approach. A set of spreadsheet end users worked with two different model-based spreadsheets. We present and analyze here the results achieved.

1 Introduction

Spreadsheets (SS) can be viewed as programming environments for non-professional programmers, the so-called "end users" [7]. End-user programmers vastly outnumber professional ones and create hundreds of millions of spreadsheets every year [10], especially for developing business applications. As numerous studies have shown, this high production rate is accompanied by an alarming high rate of errors, with some reports claiming that 90% of real-world spreadsheets contain errors [8,9].

In order to improve the robustness of SS, a considerable amount of research has been done [1,3,4,5,6]. One of the promising solutions advocates the use of a *Model-Driven Engineering* (MDE) approach, in which a business model of the spreadsheet data is defined; end users are then guided to introduce data that conforms to the model [4]. Several models have been proposed namely, templates [1], ClassSheets [3,6] and relational models [5] and also techniques to infer models from (legacy) spreadsheet data [1,3].

Although all these works claim that a MDE approach improves end-users productivity, the reality is that there is no detailed evaluation study to support this idea. In this paper, we present an empirical study that we have conducted with the aim of analyzing the practical influence that models have on productivity. We consider two different model-based SS, as proposed in [4,5]. We assess the productivity in introducing, updating and querying data in those two model-based SS and in a traditional one.

* Supported by Fundação para a Ciência e a Tecnologia, grants no. SFRH/BD/30231/2006, SFRH/BPD/46987/2008 and PTDC/EIA-CCO/108613/2008.

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 282–288, 2011.

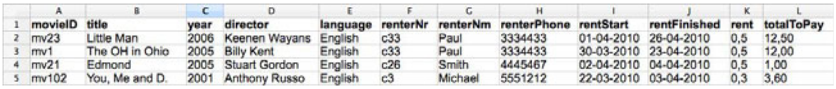
Our study is necessary and useful: it is based on a sound experimental setting which allows us to draw sound conclusions and directions for further studies on the same topic. With it, we wish to answer the following research questions:

RQ1 Do end users introduce fewer errors when they use one of the model-based spreadsheet versus the *original* unmodified spreadsheet?
RQ2 Are end users more efficient using the model-based ones?
RQ3 Do particular models lead to fewer errors in particular tasks?

We used a within subjects design, where each participant received a task list for each of three spreadsheet environments. Participants were asked to do various tasks in each spreadsheet, for example: data entry, modifications to existing data, and calculations of the data in the spreadsheet. They were encouraged to work as quickly as possible, but were not given time limits for any specific spreadsheet.

2 Model-Based Spreadsheets

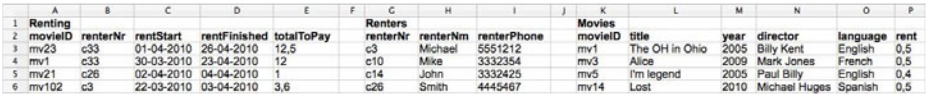
Two different techniques to tackle the problem of preventing errors in SS have been proposed [4,5]. In order to introduce them we will rely on the spreadsheet shown in Fig. 1, which represents a movie renting system (labels should be self-explicative).



	A	B	C	D	E	F	G	H	I	J	K	L
1	movieID	title	year	director	language	renterNr	renterNm	renterPhone	rentStart	rentFinished	rent	totalToPay
2	mv23	Little Man	2006	Keenen Wayans	English	c33	Paul	3334433	01-04-2010	26-04-2010	0,5	12,50
3	mv1	The OH in Ohio	2005	Billy Kent	English	c33	Paul	3334433	30-03-2010	23-04-2010	0,5	12,00
4	mv21	Edmond	2005	Stuart Gordon	English	c26	Smith	4445467	02-04-2010	04-04-2010	0,5	1,00
5	mv102	You, Me and D.	2001	Anthony Russo	English	c3	Michael	5551212	22-03-2010	03-04-2010	0,3	3,60

Fig. 1. Part of a spreadsheet representing a movie renting system

The Refactored Model: The spreadsheet in Fig. 1 shows an instance of a renting system containing redundant information: for example, client `Paul`'s information appears twice! This kind of redundancy makes maintenance complex and error-prone. A mistake is easily made (for example, by mistyping a name) and thus corrupting the data. The same information can be stored without redundancy: in the database realm, techniques for data normalization, based on the exploitation of functional dependencies (FDs) inherent in the data, are commonly used to minimize duplication of information and improve data integrity [2]. We have adapted these techniques to work with spreadsheets: from the spreadsheet data we infer a set of normalized FDs, and from them, we compute a relational model [5]. A spreadsheet respecting such model is shown in Fig. 2.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Renting						Renters				Movies					
2	movieID	renterNr	rentStart	rentFinished	totalToPay		renterNr	renterNm	renterPhone		movieID	title	year	director	language	rent
3	mv23	c33	01-04-2010	26-04-2010	12,5		c3	Michael	5551212		mv1	The OH in Ohio	2005	Billy Kent	English	0,5
4	mv1	c33	30-03-2010	23-04-2010	12		c10	Mike	3332354		mv3	Alice	2009	Mark Jones	French	0,5
5	mv21	c26	02-04-2010	04-04-2010	1		c14	John	3332425		mv5	I'm legend	2005	Paul Billy	English	0,4
6	mv102	c3	22-03-2010	03-04-2010	3,6		c26	Smith	4445467		mv14	Lost	2010	Michael Huges	Spanish	0,5

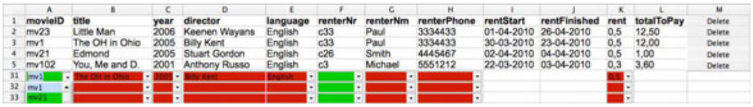
Fig. 2. Part of a refactored spreadsheet representing a movie renting system

The obtained data organization solves two well-known database problems, namely *update anomalies* and *deletion anomalies* [2]. The former occurs when we change information in one tuple but leave the same information unchanged in others, e.g., if a user changes the rent per day of `mv23` from `0.5` to `0.6`. This value occurs only once in the modular spreadsheet, where it must be updated. The latter happens when we delete a tuple and lose other information as a side effect, e.g., if a user deletes the rent in row 3 in the original spreadsheet all the information concerning movie `mv1` is eliminated.

Since we know the data relations and relationships, we can generate SS that respect them, and thus, help end users. For example, in the *renter* table, the generated spreadsheet should not allow the user to introduce two renters with the same number.

The new spreadsheet improves modularity and detects the introduction of incorrect data, and also eliminates redundancy; this should help end users commit less errors.

The Visual Model: In [4], a technique to enhance a system with mechanisms to guide end users to introduce correct data was proposed. Using the relational database schema induced by the data we construct an environment that respects that schema. For example, for the movie spreadsheet, the system must not allow the introduction of two different movies with the same number. Instead, it offers to the user a list of possible movies, such that the value to fill in the cell can be chosen. This new spreadsheet, that we show in Fig. 3, also includes advanced features which provide information to the end user about correct data that can be introduced.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	movieID	title	year	director	language	renterNr	renterNm	renterPhone	rentStart	rentFinished	rent	totalToPay	Delete
2	mv23	Little Man	2006	Keenen Wayans	English	c33	Paul	3334433	01-04-2010	26-04-2010	0,5	12,50	Delete
3	mv1	The OH in Ohio	2005	Billy Kent	English	c33	Paul	3334433	30-03-2010	23-04-2010	0,5	12,00	Delete
4	mv21	Edmond	2005	Stuart Gordon	English	c26	Smith	4445467	02-04-2010	04-04-2010	0,5	1,00	Delete
5	mv102	You, Me and D.	2001	Anthony Russo	English	c3	Michael	5551212	22-03-2010	03-04-2010	0,3	3,60	Delete
31	mv1	The OH in Ohio	2005	Billy Kent	English						0,5		
32	mv1												
33	mv21												

Fig. 3. Part of a visual spreadsheet representing a movie renting system

We consider three types of advanced features: (bidirectional) auto-completion of column values, non-editable columns and safe deletion of rows.

3 Analyzing End-Users Performance

Effectiveness. Each participant was handed 3 different lists of tasks to perform on 3 different spreadsheets, each of which constructed under a different model. We have graded their performance under each model, obtaining the results in Fig. 4. We notice that no model is neither the best nor the worst for all spreadsheets. Nevertheless, the results seem to indicate that models from [3] and [5] are not effective in reducing the number of errors, since one of them is always getting the lowest scores. This intuition deserves further development.

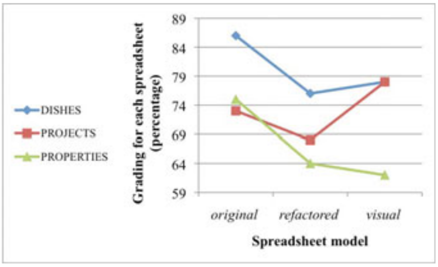


Fig. 4. Global effectiveness results

One may argue that *original* is the *model* that users are more accustomed to. Nevertheless, we remark that the more complex models *refactored* and *visual* where given no explanation; a part of our study was also to learn whether they could live on their own.

Our next step was to investigate whether the (apparent) poor results obtained by complex models are due to their own nature or if they result from participants not having understood them. In order to realize this, we studied the participations that did not achieve a score of at least 50%: 0% in *original*, 25% in *refactored* and 21% in *visual*. While in *original* no participation was graded under 50%, this was not the case for *refactored* and *visual*; this may have degraded their overall average results. For these participations, we analyzed the questionnaire that participants were asked to fill in after the session. The classifications for the post session questionnaires, for participations in the study that were graded under 50% is 24% for *refactored* and 31% for *visual*.

These results show that participants obtaining poor gradings on their effectiveness, also got extremely poor gradings for their answers to the questions assessing how they understood (or not) the models. Indeed, we can see that they were not, in average, able to answer correctly to (at least) two thirds of the questions raised in the post session questionnaire. From such results we can read that (roughly) a quarter of participants was not able to understand the more complex models used in the study, which might have caused a degradation of the global effectiveness results for these models. This also suggests that if these models are to be used within an organization, it is necessary to take some time to introduce them to end users in order to achieve maximum effectiveness.

Effectiveness by Task Type: Next, we wanted to realize how effective models are to perform the different types of tasks that we proposed: *insertion*, *edition* and *statistics*.

i) Data insertion: the *original* model was the most effective, for all 3 spreadsheets, being closely followed by *refactored* and *visual* for DISHES, and by *visual* for PROJECTS. The *refactored* model, for PROJECTS, and *refactored* and *visual*, for PROPERTIES, proved not to be competitive for data insertion. Again, we believe that this in part due to the lack of introduction to these models: the insertion of new data is the task that most likely benefits from understanding them, and also the one that can be otherwise most affected. This is confirmed by the effectiveness results observed for other task types, that we present next.

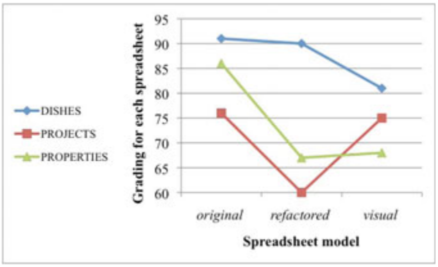


Fig. 5. Effectiveness results for insertion

ii) Data edition: once a spreadsheet is populated, we can effectively use models to edit it. This is the case of *refactored* for PROJECTS and for PROPERTIES. *original* is the most effective in editing for DISHES. *visual* is comparable to *refactored* for DISHES, but for other spreadsheets, it always achieves the lowest scores.

iii) Statistics: we can see that *visual* obtained the best results for DISHES, and that *refactored* obtained the best results for both spreadsheets PROJECTS and PROPERTIES. We can also see that all models obtained the worst results for exactly one spreadsheet.

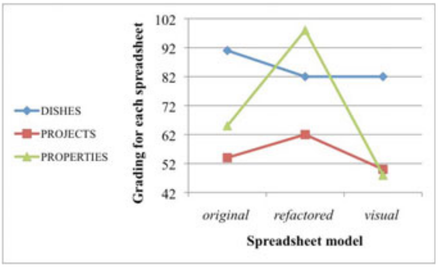


Fig. 6. Effectiveness results for edition

Results from i), ii) and iii) confirm that the models are competitive against the *original* model. On the other hand, these results allow us to draw some new conclusions: if the models are going to be used within an organization, it may not always be necessary to introduce them prior to their use. Indeed, if an organization mostly edits spreadsheet data or computes new values from such data, and does not insert new data, then the models, and specially *refactored*, may deliver good results even when they are not explicitly explained (as it was the case in our study). These results also show that it is in the data insertion tasks that the models need to be better understood by end users in order to increase effectiveness.

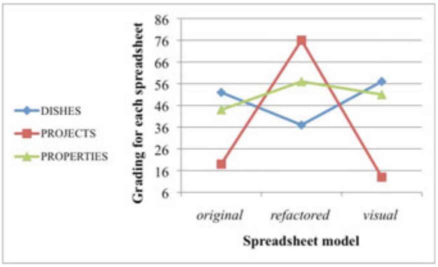


Fig. 7. Effectiveness results for statistics

Efficiency. We started by measuring, for each participant, and for each spreadsheet, the time elapsed from the moment participants started reading the list of tasks to undertake until the moment they completed the tasks proposed and moved on to a different spreadsheet or concluded the study. We are able of calculating these times by looking at the individual screen activity that was recorded during the study, for each participant: the participant stopping interacting with the computer signals the end of his/her work on a spreadsheet. The measured period therefore includes the time that participants took trying to understand the models they received each spreadsheet in. Fig. 8 presents the average of the overall times, for each spreadsheet and for each model.

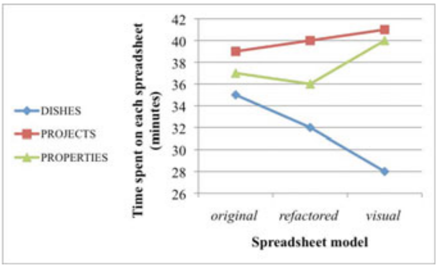


Fig. 8. Global efficiency results

We can see that *refactored* and *visual* are competitive against *original*. Indeed, participants performed fastest for DISHES in *visual*, and fastest for PROPERTIES in *refactored*. The *original* model got the best result for PROJECTS. Again, note that no introduction to *refactored* or *visual* preceded the study. Therefore, it is reasonable to assume that, for these models, the results observed include some time overhead. In an attempt to measure this overhead we extracted some more information out of the results of our study: we measured the time elapsed from the moment participants started reading, for each spreadsheet, the list of tasks to perform, until the moment they actually began editing the spreadsheet. We assume that this period corresponds exactly to the overhead of understanding each model. The average results obtained are presented in Table 1.

There is a constant average overhead of 2 minutes for almost all models and all spreadsheets, with the most significant exceptions occurring for *refactored*, for both DISHES and PROJECTS. In these cases, we can clearly notice an important time gap, which provides some evidence that *refactored* is most likely the hardest model to understand. This also comes in line with previous indications that the merits of the models can be maximized if we take the time to explain them to end users. For the particular case of efficiency, this means that the results shown in Fig. 8 could be further improved for the more complex models, and particularly for *refactored*.

Table 1. Average overhead results

	original	*refactored*	*visual*
DISHES	$2'$	$6'$	$1'$
PROJECTS	$2'$	$4'$	$2'$
PROPERTIES	$2'$	$2'$	$2'$

4 Conclusions

We have presented the results of an empirical study that we conducted in order to assess the practical interest of models for spreadsheets. From the preparation of the study, from running it and from its results, we can summarize our main contributions as follows: i) we have shown that MDE techniques can be adapted for end-users software; ii) we proved empirically that models can bring benefits for spreadsheet end users; iii) we proposed a methodology that can be reused in studies similar to ours.

Now, we seek to answer our initial research questions.

RQ1. Our observations indicate that model-based spreadsheets can improve end-user effectiveness. Even if this is not always the case, our results also indicate that deeper insight on the spreadsheet models is required to maximize effectiveness.

RQ2. We observed that, frequently, the more elaborate spreadsheet models allowed users to perform faster. Nevertheless, we were not fully able of isolating the time that participants took trying to understand the models they were working with. So, we believe that the observed efficiency results could also be better for *refactored* and *visual* if they had been previously introduced.

RQ3. Although this was not observed for inserting tasks, for editing and querying data the models did help end users. Furthermore, the results seem to indicate that the inserting data task is the one that benefits the most from better understanding the models.

With this study we have shown that there is potential in MDE techniques for helping spreadsheet end users. The study of these techniques for professional users of spreadsheets seems a promising research topic. Moreover, the use of MDE techniques in other non-professional softwares should also be investigated.

References

1. Abraham, R., Erwig, M.: Inferring templates from spreadsheets. In: Proc. of the 28th Int. Conference on Software Engineering, pp. 182–191. ACM, New York (2006)
2. Codd, E.F.: A relational model of data for large shared data banks. Communications of the ACM 13, 377–387 (1970)
3. Cunha, J., Erwig, M., Saraiva, J.: Automatically inferring classsheet models from spreadsheets. In: Proc. of the 2010 IEEE Symposium on Visual Languages and Human-Centric Computing, pp. 93–100. IEEE Computer Society, Washington, DC, USA (2010)
4. Cunha, J., Saraiva, J., Visser, J.: Discovery-based edit assistance for spreadsheets. In: Proc. of the 2009 IEEE Symposium on Visual Languages and Human-Centric Computing, pp. 233–237. IEEE Computer Society, Washington, DC, USA (2009)
5. Cunha, J., Saraiva, J., Visser, J.: From spreadsheets to relational databases and back. In: PEPM 2009: Proceedings of the 2009 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, pp. 179–188. ACM, New York (2009)
6. Engels, G., Erwig, M.: ClassSheets: Automatic generation of spreadsheet applications from object-oriented specifications. In: Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, pp. 124–133. ACM, New York (2005)
7. Nardi, B.A.: A Small Matter of Programming: Perspectives on End User Computing. MIT Press, Cambridge (1993)
8. Panko, R.R.: Spreadsheet errors: What we know. What we think we can do. In: Proceedings of the Spreadsheet Risk Symposium, European Spreadsheet Risks Interest Group (July 2000)
9. Rajalingham, K., Chadwick, D., Knight, B.: Classification of spreadsheet errors. In: European Spreadsheet Risks Interest Group, EuSpRIG (2001)
10. Scaffidi, C., Shaw, M., Myers, B.: Estimating the numbers of end users and end user programmers. In: Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing, pp. 207–214. IEEE Computer Society, Washington, DC, USA (2005)

MicroApps Development on Mobile Phones

Stefania Cuccurullo, Rita Francese, Michele Risi, and Genoveffa Tortora

University of Salerno, via Ponte don Melillo, Fisciano, Salerno, Italy
{scuccurullo,francese,mrisi,tortora}@unisa.it

Abstract. The definition of an approach supporting an End-User in the development of mobile applications is a hard task because of the characteristics and the limitations of mobile device interfaces. In this paper we present an approach and a tool to enable End-Users to visually compose their own applications directly on their mobile phone. To this aim, a touchable interface and an ad-hoc visual language have been developed, enabling the user to compose simple focused applications, named MicroApps. The user has not in charge the creation of the user interface that is automatically generated.

Keywords: Visual languages, Mobile End-User Development, Mobile Applications.

1 Introduction

The number of available applications on emerging top-of-the-range mobile devices rapidly grows, together with the increasing number of users interested in smartphones. New input sources, such as on-board camera, accelerometers, GPS, are also available. As a result, new multimodal interaction methods can be adopted, including gesture detection, device movement and context-based control [5].

The design of complex applications covering the various user demands is not an easy task. Indeed, mobile users have different preferences and employ the applications in various situations. Thus, there is the need of supporting an End-User in the composition and customization of mobile applications. However, several mobile applications can be modeled as a composition of pre-existing applications/services present on different mobile devices. In fact, these services properly handled, allow the user to define more complex applications that meet his needs. To this aim, the composition of these applications can be visually modeled through graphical symbols, associated to a particular application behavior and to a specific user interface.

This paper aims at supporting an End-User in the creation of focused mobile applications, called MicroApps. A naive End-User generates and uses a MicroApp directly on the mobile phone. A MicroApp is designed by graphically composing the functionalities offered by the various available phone applications, such as taking an image from the *Camera* object and saving it, retrieving the contacts list from the *Contacts* object, and sending an email using the *Mail* object, etc. Each object exposes a description of its user interface that is exploited to automatically generate the MicroApp interface. Moreover, the user is assisted in the composition of the functionalities provided by the mobile device. The result is a specific application based on the user needs that customizes the usage of the mobile device.

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 289–294, 2011.

This paper is structured as follows: Section 2 discusses the related work, Section 3 presents the proposed approach for the generation of mobile applications, whereas Section 4 concludes the paper.

2 Related Work

Mobile End-User development is at the beginning phase and presents new issues mainly due to the characteristics and the limitations of mobile device interfaces. The research interest towards this topic is mainly due to the growing preference revealed by the user towards the services offered by these devices and the need of customizing their applications [2].

In [4] and [5] an approach has been proposed supporting the user in the definition of context-action rules aiming at activating mobile phone functions when the rule conditions are satisfied. Differently from them, we adopt a visual interface to compose a MicroApp that can be more complex than the pattern event-action.

Jigsaw programming has been largely investigated. Program constructs are represented using icons that look like jigsaw pieces, and only icons that fit together can be composed to form legal programs [3] [6]. More recently, Google proposes AppInventor [1], that adopts a block editor to create simple programs on the PC that should be downloaded on the mobile device. This editor enables the user to program by using the OpenBlocks programming language [7]. Blocks enable the user to program repeating actions, conditions, information storing, etc. The approach we propose is very similar. It does not require the user to compose the interface and the PC usage, because MicroApps are directly created on the mobile device, and provides a form of assistance in the composition considering the compatibility of the input/output of the various actions.

3 The Proposed Approach

In this paper we focus on the creation of customized mobile applications, named MicroApp, that the user is able to compose directly on the mobile device. The framework works into two main configuration: model and enactment. In the first configuration, the user composes the application using a Visual Editor. During the enactment configuration phase a micro application is executed on the mobile device touching its icon or when a specific event occurs.

The proposed approach helps the users to manage the complexity of their activities performed with the mobile device by composing simple applications. The users do not concentrate in managing the dataflow and in the designing of the user interface, but only on the sequence of the actions needed to model the required MicroApp. In particular, the tool assists the composition by enabling the user to select an individual action from a wide range of actions available on the phone and, on the other side, it allows the user to compose the actions using an ad-hoc developed visual languages. The composition of a MicroApp is supported by a Visual Editor and a Visual Language adopting a Jigsaw-based programming approach.

In the rest of this section we describe in detail the composition and the user interface generation of a MicroApp.

3.1 MicroApp Visual Editor and Language

The behavior of a MicroApp is modeled composing its application logic through the features offered by an ad-hoc developed mobile Visual Editor that has been designed considering the limited size of the device screen.

The main idea is to eliminate all the textual components of a programming language, but providing a suitable visual language simple to use and not restricted to simple functionalities. Users can select from a wide range of actions and do not have to define dataflow among them as these aspects are automatically managed. Indeed, to enable the user to appropriately compose MicroApps, the selection of actions is supported by an underlying computational algorithms that manage the compatibility of already selected actions.

In Fig. 1 a screenshot of the Visual Editor is shown. The main area of the screen is the *Composition Area*, where the MicroApp is composed by dragging and dropping the action icon of the selected application. This area is divided in columns to allow the user sequential and parallel composition of actions. The editor allows only these types of composition rules because one of the usability objective of the proposed approach is that the language has to be easy to use and to learn and that it has to enable the composition of simple applications.

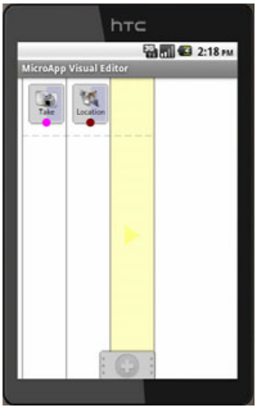
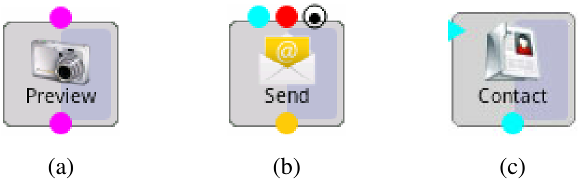


Fig. 1. The Visual Editor layout

The user adds a new action by pressing the Add button, represented by the "+" icon in the middle lower part of the Composition Area. Then, the editor opens a new window showing a list of the available applications. The list of all available applications is provided by the MicroApp Repository. When the user selects an application (e.g., *Camera*, *Contacts*, *Net* and so on), the editor shows the list of the actions available for that application. As an example for the *Camera* application the available actions are *Camera.Take, Camera.Preview,* etc. The Visual Editor assists the selection of an action by highlighting the action icons whose input is compatible with the outputs of the blocks already positioned. Once selected the action, the user clicks on a specific column of the Composition Area to add the action. The editor still assists the user disabling the columns that are not compatible with the action to be inserted. However, an action can always be inserted in the first empty column on the right, highlighted differently from the others, as shown in Fig. 1. In this case a new empty column is automatically added in the rightmost part of the editor.

Graphically, an application action available on the mobile device is represented by a rounded square containing the application icon, such as *Camera* and *Mail*, and the name of the action, such as *Preview* and *Send*, respectively. The input/output parameters are represented by colored bullets. In particular, as shown in Fig. 2, the input parameters are depicted in the higher part of the square, whereas the output parameters are shown on the bottom.

The parameters are differently colored, depending on the type of the corresponding object. As an example, the pink colored parameter represents an Image object, while the cyan colored parameter represents a Contact object, containing the contact data (e.g., name, surname, address, email, cellular phone and so on). Similarly, the red bullet represents a text string and, finally, the yellow bullet represents an email object.



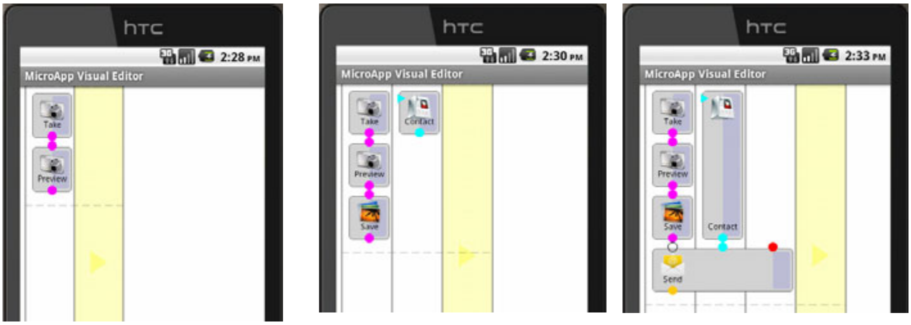
(a) (b) (c)

Fig. 2. Action block examples

Fig. 2 shows some examples of application actions. In particular, in Fig. 2(a) the *Preview* action takes as input an Image object, displays it and returns the same object as output. The generated user interface is described in the next section. In Fig. 2(b) the red bullet parameter, corresponding to a text string, is used to fill the subject field of an email. Moreover, the circled bullet denotes a variable number of parameters of any type. The objects associated to these parameters will be used to compose the email body, and could be indifferently image and/or text objects. Once the mail is sent, it will be provided as output.

Let us note that in Fig. 2(c) the *Contacts* action exposes a triangle parameter in the left hand side. This kind of parameter has to be assigned during the application composition, and the associated value will be fixed for each execution of this MicroApp. This means that at design time the user has to select a contact present in the contact list.

Fig. 3 shows some composition rules available to create a MicroApp. In particular, Fig. 3(a) depicts an example of a successful sequential action flow. In fact, the output parameter of the topmost action is compatible with input parameter of the lower action. Fig. 3(b) shows an example of a parallel action flow, whereas Fig. 3(c) depicts an example of joint action flow. In particular, a new action is added to collect the outputs of the actions in the first and second columns. The user initially drags and drops the action on a particular column (i.e., the first column), and successively clicks on the other columns for associating the input parameters to the action, respecting the number and the type of the inputs. Note that if, as in the case of the second column, there is an empty space, the action *Contacts* is automatically lengthened. If the user provides no action in the third column, the text input will be provided by the default action associated to the text parameter (i.e., the *Text.Input* action), otherwise only an action with a textual output parameter should to be added successively.



(a) (b) (c)

Fig. 3. Example of composition rules

Once the user terminates the MicroApp modeling phase, he selects the *Deploy* command that saves an XML description of the composition process in the MicroApp Repository, ready to be enacted by the MicroApp Engine.

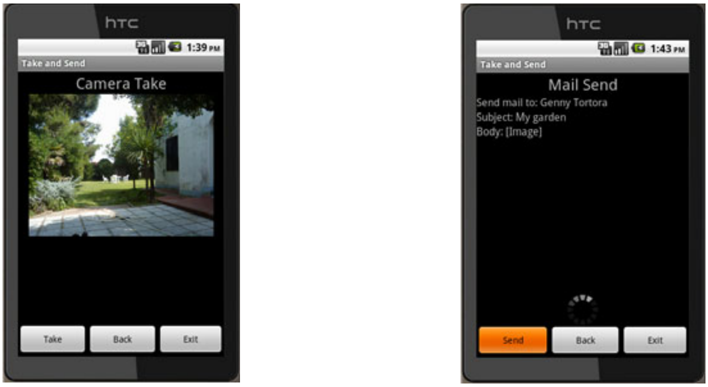
When the MicroApp Engine loads the XML description, it translates the description into an execution sequence by instantiating the action objects, and then running the process over them.

3.2 The MicroApp User Interface

The composition of the user interface is not an easy task for a non programming user. Indeed, there is the need of model the user interaction in terms of GUI elements, such as windows, pull-down menus, buttons, scroll bars, iconic images, wizards, etc.

In this paper each application action is annotated by a user interface. The MicroApp user interface is automatically generated by combining the annotated actions presentation frontends. When generating a MicroApp, the interface is created following the Microsoft PowerPoint presentation approach: each action is presented as a slide.

As an example, Fig. 4(a) shows the interface generated for the *Camera.Take* action. In particular, the user is able to get an image, touching the *Take* button, or he can go *Back* in the control flow or eventually *Exit* the MicroApp by pressing the appropriate buttons. Similarly, Fig. 4(b) shows the user interface for the *Mail.Send* action. In particular, a preview of the mail is provided to the user that sends the mail by touching the *Send* button.



Send mail to: Genny Tortora
Subject: My garden
Body: [Image]

(a) (b)

Fig. 4. The user interface of the *Camera.Take* and *Mail.Send* actions

4 Conclusion

In this paper we have presented a mobile application that supports the user in the visual composition of customized applications for mobile devices. It has been designed for naive users and does not require the user involvement in the specification of the user interface.

The approach has been implemented in a prototype of a supporting users in the composition of MicroApps. In particular, the prototype is running on an Android based HTC device, by using the SDK version 2.2.

References

1. AppInventor, Google, `http://appinventor.googlelabs.com/about/` (retrieved on November 2010)
2. Danado, J., Davies, M., Ricca, P., Fensel, A.: An authoring tool for user generated mobile services. In: Berre, A.J., Gómez-Pérez, A., Tutschku, K., Fensel, D. (eds.) FIS 2010. LNCS, vol. 6369, pp. 118–127. Springer, Heidelberg (2010)
3. Glinert, E.P.: Out of Flatland: Towards 3-D Visual Programming. In: Proceedings of FJCC 1987-Fall Joint Computer Conference, October 25-29, pp. 292–299. IEEE Computer Society, Dallas (1987)
4. Häkkilä, J., Korpipää, P., Ronkainen, S., Tuomela, U.: Interaction and End-User Programming with a Context-Aware Mobile Application. In: Costabile, M.F., Paternó, F. (eds.) INTERACT 2005. LNCS, vol. 3585, pp. 927–937. Springer, Heidelberg (2005)
5. Korpipää, P., Häkkilä, J., Malm, E.-J., Rantakokko, T., Kyllönen, V., Kela, J., Känsälä, I., Mäntyjärvi, J.: Customizing User Interaction in Smart Phones. IEEE Pervasive Computing: Mobile and Ubiquitous Systems 5(3), 82–90 (2006)
6. Osawa, N.: Jigsaw-Puzzle-Like 3D Diagrams to Represent Syntactical Constraints. In: Sixth International Conference on Information Visualisation, IV (2002)
7. Roque, R.: OpenBlocks: an extendable framework for graphical block programming systems, Master Thesis Massachusetts Institute of Technology (2007), `http://dspace.mit.edu/handle/1721.1/41550`

Playbook: Revision Control and Comparison for Interactive Mockups

Stephen Oney[1], John Barton[2], Brad Myers[1], Tessa Lau[2], and Jeffrey Nichols[2]

[1] Carnegie Mellon University
5000 Forbes Ave. Pittsburgh, PA 15213
{soney,bam}@cs.cmu.edu
[2] IBM Almaden Research Center
650 Harry Rd. San Jose, CA 95120
{bartonjj,tessalau,jwnichols}@us.ibm.com

Abstract. When designing interactive interfaces and behaviors, interface designers compare and contrast multiple design ideas, often creating and testing many intermediate user interface prototypes before deciding on a final design. However, existing interface prototyping and creation tools do not effectively let designers explore, compare, or keep track of older versions of interface mockups, implicitly making the assumption that the users of these tools will work with one design alternative at a time. To explore how to enable designers to work with multiple designs in a prototyping tool, we created Playbook, a new system oriented towards helping interface designers keep track of, compare, and create interactive mockups. In this paper, we describe Playbook and discuss ways that future prototyping tools can better support the workflow of designers.

Keywords: prototyping, mockups, interface design, versioning.

1 Introduction

In the intermediate stages of interface design, designers typically produce a large number of design artifacts: site maps, story boards, static mock-ups, interactive prototypes, etc. [1]. Although a number of surveys and empirical evidence have shown that designers need better tools to manage and evaluate these design artifacts [1][2][3], designers are still using ad-hoc versioning solutions, like manually renaming files [1][4]. This may be for two reasons: first, revision control systems do not effectively fit in with interface designers' workflows or the tools they most frequently use for creating mockups. Second, while methods for keeping track of and comparing static artifacts, like site maps, are relatively straightforward, there are no appropriate methods for keeping track of *interactive* artifacts, like interactive mockups, and this presents a significant research challenge.

Interactive artifacts are ill suited for tracking with traditional revision control methods because they are defined by both their appearance and behavior. Although many imperative languages conflate the two, interface designers should ideally be able to define and modify each independently. Having separate revision control repositories for appearance and behavior is not practical because the two aspects are

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 295–300, 2011.

interdependent. Particular revisions of the interface behavior are only compatible with a subset of the versions of the interface appearance because, for example, the code for the behavior will be dependent on the presence of specific interface elements in the appearance. Thus, a revision control system for interactive prototypes should be able to keep track of not only revisions of appearances and behaviors, but also compatibility between the two.

In addition to keeping track of revisions, a revision control system should enable comparisons to be made across different revisions. For static artifacts, this is largely a solved problem, as evidenced by the large number of textual and image-based difference systems. For interactive artifacts, however, comparisons between mockups are more difficult to make in a meaningful way, beyond changes in appearance, or beyond textual differences in the code responsible for the behavior.

We created Playbook to explore solutions to the aforementioned issues of revision tracking for interactive interface mockups. To manage revisions of both appearance and behavior, Playbook keeps track of layered images that define the appearance of the mockup and "scripts" that define the behavior of the mockup. To effectively fit in with interface designers' existing workflows and tools, Playbook allows designers to upload layered images quickly and directly from Photoshop. To add interactivity to these static layered images, Playbook uses a scripting language inspired by that of CoScripter [5], where high-level scripts describe the behaviors of a mockup. Our scripting language describes behaviors on a sufficiently high level that one script may be applied to mockups with very different appearances. Playbook scripts are grouped around the specific "tasks" that they enable the user of the interface to perform. For example, one task for a mockup of a movie rental website might be "rent a movie." Scripts grouped under this task describe how different interactive mockups react as the user goes through the steps of renting a movie. If the interface changes dramatically between revisions, very different scripts may be required to describe how the interface behaves when the user is performing a task. To allow interactive mockups to be compared, Playbook allows for mappings between these scripts within a task to define equivalent parts of different scripts, as is shown in Figure 1. This paper demonstrates that design tools can help designers manage and compare different revisions of interactive prototypes and presents Playbook, a tool for creating interactive prototypes that embodies this idea.

2 Related Work

A number of tools have been created to enable the creation of interactive mockups of various fidelities, including Adobe's Flash Catalyst, DENIM [6], and Designer's Outpost [7]. However, our focus in Playbook is on how to enable revision tracking and comparison, rather than on how the interactive behaviors are defined originally. Playbook is only concerned with the language describing mockup behavior to the extent that it is simple enough to be used by interface designers without a programming background and allows for revision control and comparison between interactive mockups.

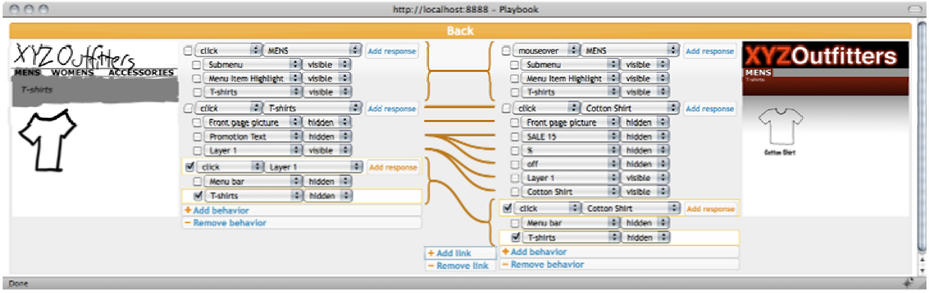
Other systems have been built with the intention of exploring and comparing designs. Cogtool [8] supports estimating expert performances on mockup user interfaces and displays a grid of interface designs along with their performance in doing user-specified tasks. Whereas Cogtool allows interface designers to compare prototypes using task completion times, Playbook allows designers compare different prototypes more qualitatively by being able to interact with the prototypes side-by-side. Juxtapose [9] is another system that lets designers compare different possible designs side-by-side. However, Juxtapose only allows low-level user input events to be replicated, such as mouse clicks at a particular (x,y) location on the screen, when comparing different prototypes. This limits the differentiation that is permissible when comparing interactive mockups side-by-side with Juxtapose.

3 Design

Playbook mockups start out as layered Photoshop images. Each interactive element of the interface must be in a separate layer. When the designer has a mockup they are happy with, the next step is to upload that mockup to the Playbook server, which keeps track of every uploaded revision. To make this step as simple as possible, we created a Photoshop plugin that adds a menu item to the Photoshop interface's File menu that does this. After the user clicks this menu item, the file is uploaded and the user's web browser is opened, pointing to the Playbook web page, so it operates like a versioned save feature.

After uploading the layered Photoshop file to the Playbook server, the next step is to create scripts to make the mockups interactive. Every script is then classified under a "task" group, which describes, on a high level, what the scripts in that group enable the user to do on the mockup. For example, in a mockup for a clothing website, one could write a set of scripts for the task of buying a t-shirt. Every script in that task group would describe how a user would buy a t-shirt in a particular version of that mockup (click the 'mens' button, and then the menu overlay should appear…click the 't-shirts' button, and a list of t-shirts should appear, etc.). These scripts can be thought of as interaction traces through a mockup. Every script is a set of "behaviors" which consist of one "stimulus" and any number of "responses." A stimulus is a user action to which the mockup will respond. For example "mouseover the 'womens' layer" or "click the 't-shirt' layer." Each response is a simple reaction to a stimulus. Only two responses are currently supported: hiding and showing layers. While the set of responses is limited, evidence from popular prototyping tools such as Balsamiq shows that even with these simple responses, designers can mock-up many of the desired behaviors that appear in web interfaces.

When writing a script for the first time, every behavior is specified by the designer. Designers can write these behaviors by demonstrating the stimuli on the interface mockup, and can later go back and edit these scripts manually if necessary. As the user demonstrates an action, the currently-selected behavior in the script updates itself by setting its stimulus to the action the user just performed. For example, if the user demonstrates a click on a particular layer, the currently-selected behavior's stimulus is set to clicking on that layer. The stimulus options for behaviors are: mouseover, mouseout, mousedown, mouseup, and click. Playbook does not do any inferencing or reasoning on the stimulus-response pairs the user writes.



□ click MENS Add response
- □ Submenu visible
- □ Menu Item Highlight visible
- □ T-shirts visible

□ click T-shirts Add response
- □ Front page picture hidden
- □ Promotion Text hidden
- □ Layer 1 visible

☑ click Layer 1 Add response
- □ Menu bar hidden
- ☑ T-shirts hidden

+ Add behavior
− Remove behavior

+ Add link
− Remove link

□ mouseover MENS Add response
- □ Submenu visible
- □ Menu Item Highlight visible
- □ T-shirts visible

□ click Cotton Shirt Add response
- □ Front page picture hidden
- □ SALE 15 hidden
- □ % hidden
- □ off hidden
- □ Layer 1 visible
- □ Cotton Shirt visible

☑ click Cotton Shirt Add response
- □ Menu bar hidden
- ☑ T-shirts hidden

+ Add behavior
− Remove behavior

Fig. 1. The user is mapping stimuli and responses from the script on the left to the script on the right. Equivalent objects have a line between them. For example, a click on the "MENS" layer in the left mockup is equivalent to a mouseover of the "MENS" layer in the right mockup. When whole behaviors are equivalent, Playbook uses curly braces and a single line between the behaviors to reduce clutter. The mappings between mockup stimuli and responses are used to allow the user to interact with multiple interactive mockups simultaneously.

Every script is tied to a particular mockup (but may be used across versions of that mockup), because it uses layers that may only be in that mockup. However, we wanted to give the designer the ability to easily apply old scripts to new mockups. For example, if a designer writes a script for version 1 of a mockup, and makes some minor tweaks to the graphics between version 1 and version 2, we did not want the designer to have to rewrite the scripts that they wrote for version 1. Thus, after the user writes a script, Playbook automatically tries to apply the script to new versions of that mockup, and generates scripts for them. This is done by trying to replicate all of the behaviors from the previous script by looking for layers with the same name on the new version. If a behavior, or part of a behavior, refers to a layer name that is not in the new version, Playbook omits that part of the behavior. The designer must then verify these newly generated scripts before they can be used. As they are verifying the script, they can make any desired changes to its content.

One of the novel features of Playbook is the interface "compare" feature, which allows designers to compare interactive versions of their mockups by interacting with both simultaneously. We designed the compare feature to allow mockups with very different user interfaces to be compared. Allowing the designer to specify which actions are 'equivalent' on different interfaces enables this. When designing the compare feature, the first important design issue that needed to be addressed was: at what granularity should designers be able to specify equivalence: complete behaviors or individual stimuli and responses? We decided to allow designers to specify equivalencies of stimuli and responses because it increased the flexibility of what could be equivalent. Between interfaces, stimuli can be marked as equivalent to other stimuli, and responses can be marked as equivalent to other responses in a many-to-many relationship (any number of stimuli or behaviors can be marked as equivalent to any number of stimuli or behaviors in another mockup).

Another design issue was how and when designers would view and edit the equivalency relationships. We believe that the most natural way of visualizing equivalency relationships is by drawing lines between equivalent stimuli and responses, as shown in the center of Figure 1.

These equivalency relationships between actions can only be specified in scripts within a task. For example, the ‘buy a t-shirt’ script in mockup version 1 can only have equivalency mappings with the ‘buy a t-shirt’ script in mockup version 2; mappings cannot be made across tasks. When Playbook generates a script for a new mockup based on an old script, it automatically generates a set of equivalency relationships that can be verified, discarded, or augmented by the designer.

The equivalency connections are the basis for how Playbook allows the user to interact with multiple prototypes at the same time. When the user performs an action (stimulus) on one interactive mockup, Playbook then looks for the equivalent stimulus on any other mockups that are running. If there is an equivalent stimulus, Playbook simulates the stimulus on that prototype. If not, Playbook looks at the responses for the original stimulus. For every response, Playbook looks for equivalent responses in the other mockup, and simulates the stimulus responsible for that response. If the layer responsible for that stimulus on the equivalent prototype is not visible, Playbook still executes the stimulus, giving the designer a warning.

One of the benefits of Playbook being a web platform is that it enables multiple people to easily share the interactive prototypes. Playbook provides menu items to allow users to download a previous mockup as a Photoshop file, make changes to the mockup through Photoshop, and re-upload it to the Playbook server as a new version. Further, Playbook generates small HTML snippets that allow these interactive mockups to be easily embedded into other webpages. One could, for example, embed an interactive mockup into a wiki page that describes the interface and use the interactive mockup as a working example.

4 Implementation

Because it is a web-based interface, Playbook was implemented almost entirely in JavaScript. Using Photoshop’s built-in JavaScript plugin capability, we added an “Upload to Playbook server” menu item to the Photoshop File Menu. On the Playbook server, the Photoshop file is processed, splitting the layers into separate image files. The Playbook server runs a copy of Photoshop and uses another Photoshop script to create the web version of the layer image so it can be used by the mockup.

The Playbook server stores all of the information it contains about each mockup, layer, script, etc. in a database. Playbook’s web interface, in turn, periodically updates itself by querying the database. This allows users to stay updated if other team members are using the Playbook interface at the same time. The web interface also periodically saves any changes that are made to scripts back to the database, so that the users’ scripts will still be on the server after they leave the page.

5 Conclusion and Future Work

This paper presented Playbook, a system that allows designers to maintain and compare multiple revisions of interactive prototypes. Playbook is a proof-of-concept that shows that it is feasible to enable revision control and comparison of interactive mockups while working with the tools that interface designers currently use. We believe that many of the ideas behind Playbook can be used to augment future prototyping tools, including giving designers the ability to keep track of old designs and design alternatives, and allowing designers to compare prototypes. For future research, we plan to explore alternative ways to highlight differences between interactive mockups, expand the range of what can be prototyped using our scripting language without raising the floor of knowledge required, and explore ways to use Playbook as a "boundary object" to improve communication between designers and developers.

Finally, while Playbook is a system especially for interface designers, we also plan on exploring ways of applying some of the principles behind Playbook to development and prototyping systems for End User Development. To the extent that these systems permit separation of concerns between appearance and behaviors, augmenting them with some of Playbook's features may enable end users to better explore their design space and create more thoroughly designed artifacts

Acknowledgments. We thank our collaborators at IBM and the conference organizers and reviewers for their helpful advice. In addition, we thank the ARCS and Ford foundations for their generous support. This research was also partially supported by the National Science Foundation under grants IIS-0757511 and CCF-0811610.

References

1. Ozenc, F.K., Kim, M., Zimmerman, J., Oney, S., Myers, B.: How to support designers in getting hold of the immaterial material of software. In: ACM CHI Conf., pp. 2513–2522 (2010)
2. Grigoreanu, V., Fernandez, R., Inkpen, K., Robertson, G.: What designers want: Needs of interactive application designers. In: IEEE VL/HCC, pp. 139–146 (2009)
3. Newman, M.W., Landay, J.A.: Sitemaps, storyboards, and specifications: a sketch of Web site design practice. In: Proceedings of the 3rd Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques (DIS), pp. 263–274 (2000)
4. Carter, A., Hundhausen, C.: How is User Interface Prototyping Really Done in Practice? A Survey of User Interface Designers. In: IEEE VL/HCC, pp. 207–211 (2010)
5. Leshed, G., Haber, E.M., Matthews, T., Lau, T.: CoScripter: automating & sharing how-to knowledge in the enterprise. In: ACM CHI Conf., pp. 1719–1728 (2008)
6. Lin, J., Thomsen, M., Landay, J.: A visual language for sketching large and complex interactive designs. In: ACM CHI Conf., pp. 307–314 (2002)
7. Klemmer, S.R., Newman, M.W., Farrell, R., Bilezikjian, M., Landay, J.A.: The designers' outpost: a tangible interface for collaborative web site. In: ACM Symposium on User Interface Software and Technology (UIST), pp. 1–10 (2001)
8. Hudson, S., John, B., Knudsen, K., Byrne, M.: A tool for creating predictive performance models from user interface demonstrations. In: ACM Symposium on User Interface Software and Technology (UIST), pp. 93–102 (1999)
9. Hartmann, B., Yu, L., Allison, A., Yang, Y., Klemmer, S.R.: Design as exploration: creating interface alternatives through parallel authoring and runtime tuning. In: ACM Symposium on User Interface Software and Technology (UIST), pp. 91–100 (2008)

Expressing Use – Infrastructure Probes in Professional Environments

Jan Hess[1], Christian Doerner[2], Volkmar Pipek[1], and Torben Wiedenhoefer[1]

[1] University of Siegen
{jan.hess,volkmar.pipek,torben.wiedenhoefer}@uni-siegen.de
[2] Carnegie Mellon University
cdoerner@cs.cmu.edu

Abstract. Cultural Probes have proven to be a successful approach for involving end users in exploring the context one might design for. Several studies made value of probes in domestic contexts to inspire the design of systems but the role of probes in business contexts is underexplored. In this paper we report on our experiences of adapting probes to be used as a method for a user-centered design process in work environments. Our probes, called Infrastructure Probes, are tools for self-documentation and reflection to enable employees to document usage, problems and suggestions related to their IT and workplace infrastructure. We evaluated the Infrastructure Probes in two field studies. In this paper we motivate the approach, discuss values and issues by introducing probes into a business context and reflect on the lessons we learnt.

Keywords: Human Factors, Empirical Study, Probes, Workplace Infrastructure, Field Research, Participatory Design.

1 Introduction

In previous works [2] we focused on the design of software systems for Small and Medium-sized Enterprises (SME) which can easily be adapted by users themselves. We were interested in the existing practice of end-user activities, e.g. in relation of how they adapt the IT infrastructure, how they report and resolve technological problems and how they communicate work arounds. The practices we were interested in are incident-based ones, only weakly routinized and regarded as peripheral to the 'actual, productive work'. Even if ethnographic methods are being widely used in user-centered design processes to provide a rich picture of the work environment, the contextualized feedback we are interested in is difficult to explore this way.

Apart from ethnographic methods (e.g. observations, surveys, interviews) "Cultural Probes" [3] has gained scientific interest within the past decade. The approach introduced by Gaver et al. and in meantime adapted in several studies [1], can be characterized as an explorative self-reflection method that enables users to provide open and creative forms of feedback from the context in question. Probes usually consist of several tools that enable participants to capture their environment and also to express their feelings and wishes, leading to a documentation of their context. These properties

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 301–306, 2011.

make probes unique in a sense since they easily allow capturing information about the researched context without distracting people too much in their environment.

Although probes were introduced to inspire the design in domestic contexts, we were interested to see if they can also be adopted for professional settings to get insights into work contexts. There are a few studies of probes for business contexts. Jäsköö and Mattelmäki [5] adapted the probes concepts to gain an understanding of routines and actions of nurses. Lucero and Martens [7] used probes as a first ethnographical part for identifying design activities that can be supported by mixed reality. In [8] Lucero and Mattelmäki describe an approach they called 'Professional Probes'. In their work they reflect their findings gathered with industrial designers. They spent a considerable amount of work and resources in creating their probes. However, they identified several problems of using probes in a professional context. Several participants dropped the study while mentioning a lack of time for the documentation process. For some attendees the probes study also turned into an obligation they had to do.

In our research we also adapted probes to a business context. While conducting standard ethnographical methods for the overall research project [2] we experimented with the probes in addition to gather own hands-on experience and to put the focus on activities related to end-user development (EUD) [6]. Our work is motivated by the question to what extend a probes design can support empirical exploration of the current EUD practice. Such practice, e.g. customize a module to reach a specific goal, would not only be of interest for us as researcher but also of interest for the employees within the same company, e.g. by sharing such documentations with each other. In order to reach those goals, our probes approach is a combination of physical artifacts, such as cameras and a 'Technology Probe' [4] which we realized as a snapshot tool. Such a technological probe combines "… the social science goal of collecting information, the engineering goal of field-testing the technology, and the design goal of inspiring users and designers…" [4]. We studied the Infrastructure Probes in a real world context, by involving five small- and medium-sized companies in our study. Based on this broad practical setting, we are able to report on the lessons we learned and will also draw some valuable conclusions from our work.

2 Concept

Methods for End-User Development (EUD) enable end-users to get actively engaged in adaption and development of information systems [6]. Our so-called Infrastructure Probes (IPs) are intended for self-documentation and reflection on problems and use innovations in the everyday practice of the employees. The IPs should enable participants to help each other and to improve their working infrastructure. Since users may be good at solving the problems they have, but not at documenting how they did it, the Infrastructure Probes should help them to simplify this process. The Infrastructure Probes help users to structure their documentations, making it easier for others to understand them. The arrangement of the IPs targets the documentation of usages of IT infrastructures. Their design is theoretically informed by research on E-Infrastructures as described in [10]: The IPs specifically aim at documenting 'infrastructure breakdowns' and 'use innovations'.

Our Infrastructure Probes are an arrangement of different probes/tools to enable users to “self-document” their environment (see Figure 1). The IP package should attract users in different ways depending on their skills and knowledge. All probes are quite simple to understand, making sure to get as many users involved as possible. The following collection of probes is contained in the IPs package: A *digital camera* (Figure 1, #2) can be used to reveal problems that are not restricted to software alone (e.g. if the transfer of data between two applications is done by paper documents). The *Post-its* (Figure 1, #3) can be used to take down short notes or to “highlight” specific things in a picture. The *forms* (Figure 1, #4) are designed for a structured description of problems and problem solving strategies. The *IT diary* (Figure 1, #5) has two functions: First, it offers an unstructured way to document problems and problem solving strategies. Second, it allows participants to put documentations which have been made with different probes in a connected, chronological order. The *writing pad* (Figure 1, #6) can be used for the creation of paper mock-ups. We also added a *user manual* (Figure 1. #7) that describes the function and possible usage of each probe. Our ‘Technology Probe’ [8] – a *snapshot tool* (installed on the USB-Stick shown in Figure 1, #1) – is considered to be the most important probe of the package because it gives users the chance to create, annotate and manage screen shots. The annotation of the screen shots is important, as it allows users to provide more detailed context information about the problem at hand.

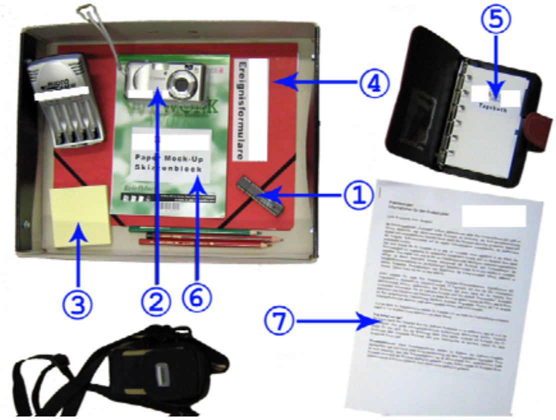


Fig. 1. Infrastructure Probes

3 Evaluation

In the first evaluation of the Infrastructure Probes we basically aimed to answer the following questions. First, what is the general perception of the usefulness of the probes in the work context of each participant? Second, what kind of problems did the participants record? Third, in which way were the probes used? Forth, how can the quality of the problem descriptions be evaluated and fifth, how usable are the probe tools. We created twelve IPs packages with the previously described set of different probes. The packages were given for eight weeks to twelve participants working for five different SME. We gave each participant a short oral introduction, telling her or him about the aim of IPs and about the possible usage of each probe. After the first third of the eight-week trial, we interviewed the participants via telephone to get first impressions about their experiences with the IPs. At the end of the trial, we analyzed the data and organized feedback workshops together with the participants to discuss the results and ask them about their experiences with using the probes.

For the second evaluation we gave eleven participants an improved version of the IPs. Regarding the IT-diary, the Post-it's and the forms, participants from the first trial noted that these probes were too bureaucratic and required too much time to be used properly. For these reasons we didn't use these probes in the second evaluation. The digital cameras from the first phase were used again. One of the major improvements concerned the snapshot tool. According to the suitability of the users' tasks, we integrated an email function which enables users to send screenshots to other persons. Collaboration among employees would be possible this way, e.g. by tailoring artifacts [9] and documenting of the adaption. This time, only four of the five companies of the first evaluation participated. Based on our experiences, we demonstrated the use of the snapshot tool and the other tools during the introduction and also gave participants the chance to try each of the probes. After several months we end up the evaluation by interviewing nine participants separately to get more detailed information about their experiences with the probes.

Seven out of 11 participants used the IPs in the first evaluation. Table 1 provides an overview of the collected data. Instead of using our tools, three participants used an alternative documentation method. They took screenshots and copied and pasted them into Word documents. While the snapshot tool was used by some of the participants, the IT diary and the writing pad were not used by anyone and only two persons used the forms.

Table 1. Quantitative overview of the feedback (*notes include forms)

Feedback	Quantity
Pictures taken by the camera	26
Screenshots taken by the snapshot tool	11
Screenshots embedded in WORD documents	17
Handwritten notes*	5

The camera was primarily used to document participants' workspace. In addition, one of the participants used the camera to take photos of her screen. Her pictures showed different dialogs which were related to driver problems and error messages. She also tried to describe her problems by using the forms. Four persons used the screenshot tool as intended by us, providing a rather rich documentation. Three of these participants worked for the same company where a culture of documenting problems and solutions (with another screenshot tool) was already established. These three participants made screenshots of the applications and used the tool's annotation functions to describe their problems. For example, one of the participants took a screenshot of an order document in his SAP system and described it as *"Transfer of supplier master data"*. He also added a complaint: *"[...] the current master data of* *the supplier is obviously not transferred"* by using copy-and-paste which could lead to a wrong address on the order document. In the comment field of another screenshot he extensively described problems that can occur in the case an article is "locked".

The second phase with the modified approach did not lead to the expected adoption of the IPs. Instead of using the Infrastructure Probes, one participant used Microsoft Excel to describe his problems. Within his Excel sheet, we found problem descriptions with Microsoft Office applications, general software errors and difficulties with SAP software modules. Another participant, working in quality management, claimed that they already use a snapshot and reporting tool to indicate and describe product shortcomings and problems.

To get detailed information about the acceptance/non-acceptance of the IPs and to reflect on the method, we conducted telephone interviews and feedback workshops. The majority of the participants said that they did not use the probes because they did not have enough time during the day. The use of the probes seemed to be too time consuming and too difficult to incorporate into the daily work routine for most participants. In case problems occurred, participants stayed focused on the resolutions of these problems and did not think about using the Infrastructure Probes to document these processes. Another important aspect for not using the probes was the fact that most users considered the IPs as a "job" which had to be done in addition to their regular work and not as a useful extra task that could stimulate collaboration to improve the IT and workplace infrastructure.

Technical problems also lead to the fact that participants refused to use the probes. The participants from one firm did not use the snapshot tool because of policy constraints from the IT department. In the first evaluation two participants also had problems in using the USB sticks with the snapshot tool. However, according to the majority of the participants, the snapshot tool had been the most interesting tool in the probes package. In the feedback workshops, participants suggested improvements for the snapshot tool. They showed us typical work processes where they printed documentations or help instructions that were kept in folders. To enrich these documentation processes, a print functionality should be included in the snapshot tool as well as the option to create a series of screenshots. Additionally, a faster and easier-to-use interface was strongly demanded. Especially for users with less technical knowledge the tool was not as easy to use as it should be.

4 Conclusion

Our research was motivated by the necessities to find efficient means to capture an incident-oriented, weakly routinized and peripheral work practice (coping with workplace infrastructure breakdowns and innovations). Participants who used the Infrastructure Probes gave us concrete examples of breakdown situations which we could discuss later on in more detail. These examples were helpful for us because we could not identify them in the interviews that we had conducted before. The method worked – at least for some users – as intended by us. Participants informed us about their work environment and problems with the IT infrastructure. From this point of view, the Infrastructure Probes can add value to other empirical methods. However, in our evaluation of the Infrastructure Probes we also identified different aspects that make it hard to use them in business contexts. For the participants it was difficult to integrate them in their work practice, time constraints also did not gave enough space to use the probes as intended by us.

The Cultural Probes are well-designed artifacts that stimulate use. Lucero & Mattelmäki [8] recommended adapting them to a fluid and playful process to avoid obligation. In contrast to this, our Infrastructure Probes are less playful, although we tried to integrate some 'funny' things in the packages of the second study, such as comics, mouse pad, emoticons within the snapshot tool. Maybe we would have gotten better results if the probes had been more attractive, for example by using better designed material to stimulate creativity. However, the Infrastructure Probes needed to be balanced out between a creative or even playful [8] motivation for using them (that still has to done by further improvements of the design) and a 'serious' motivation of getting something in return (e.g. help, documentation of problem solutions). In addition, we consider the first confrontation of users with the Infrastructure Probes is a critical point for adoption. The reason is that we saw the strongest interest of participants in the first evaluation which means that the initial try-out experience of participants is very important. In the long run, the fact of having a personal benefit from using the probes becomes more important.

Acknowledgments. This research was supported by the German Federal Ministry for Education and Research in the project EUDISMES.

References

1. Boehner, K., Vertesi, J., Sengers, P., Dourish, P.: How HCI Interprets the Probes. In: Proceedings of the CHI 2007, pp. 1077–1086 (2007)
2. Doerner, C., Hess, J., Pipek, V.: Improving information systems by end user development: A case study. In: Proceedings of the 15th European Conference on Information Systems (ECIS 2007), St. Gallen, Switzerland, pp. 783–794 (2007)
3. Gaver, B., Dunne, T., Pacenti, E.: Design: Cultural probes. Interactions 6(1), 21–29
4. Hutchinson, H., Mackay, W., Westerlund, B., Bederson, B.B., Druin, A., Plaisant, C., Beaudouin-Lafon, M., Conversy, S., Evans, H., Hansen, H., Roussel, N., Eiderbäck, B.: Technology probes: inspiring design for and with families. In: Proceedings of CHI 2003, pp. 17–24. ACM Press, New York (2003)
5. Jääskö, V., Mattelmäki, T.: Observing and probing. In: Proceedings of the DPPI 2003, pp. 126–131. ACM Press, New York (2003)
6. Lieberman, H., Paternò, F., Wulf, V.: End User Development. Springer, Dordrecht (2006)
7. Lucero, A., Martens, J.-B.: Supporting the creation of Mood Boards: Industrial Design in Mixed Reality. In: The First IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TableTop 2006), pp. 125–126 (2006)
8. Lucero, A., Mattelmäki, T.: Professional Probes: A Pleasurable Little Extra for the Participant's Work. In: Proceedings of the 2nd IASTED, pp. 170–176 (2007)
9. Pipek, V., Kahler, H.: Supporting collaborative tailoring. In: Lieberman, H., Paterno, F., Wulf, V. (eds.) End-User Development. Springer, Berlin (2006)
10. Pipek, V., Syrjänen, A.-L.: Infrastructuring as Capturing In-Situ Design. In: 7th Mediterranean Conference on Information Systems. Association of Information Systems (2006)

From Top to Bottom: End User Development, Motivation, Creativity and Organisational Support

Patrick Kierkegaard and Panos Markopoulos

Eindhoven University of Technology,
Department of Industrial Design,
P.O. Box 513, 5600 Eindhoven, The Netherlands
{p.m.kierkegaard,p.markopoulos}@tue.nl

Abstract. This paper examines the socio-technical and organisational factors that influence the adoption of End User Development (EUD) technology and practices at the workplace. This research focuses on the rehabilitation industry, where a 64-item paper-based survey was completed by 52 therapists working at two rehabilitation clinics in the Netherlands. Results suggest that therapists need to be motivated to act as creators of technological innovations in rehabilitation as this is not part of their current work culture and current rewards for such innovations are perceived as small. Most likely incentives for therapists are (1) time allocated for end user development practices (2) monetary compensation for overtime required for this work (3) and intellectual ownership of the innovation.

Keywords: organizational studies, intrinsic motivation, extrinsic motivation, therapy, technology acceptance, creativity, end-user development.

1 Introduction

Theoretical and empirical works in the field of End User Development have often examined the cognitive aspects of programming. Comparatively, much less attention has been placed on the motivational, socio-technical, and organisation factors that influence an individual's and especially an employee's willingness to successfully engage in EUD. A question that is critical to answer for the eventual feasibility of EUD as a software creation approach pertains to what are the social and organizational conditions that will enable the adoption and success of EUD.

Little is known on how an organisation can best foster and support the employee's acceptance and use of EUD technology. The research described here is part of an investigation processes for introducing novel technologies in rehabilitation. Specifically, it aims to provide EUD technologies for therapists so that they can create their own tangible and robotic interaction solutions for supporting patient therapy. EUD can potentially overcome the difficulty for technology developers to access highly specialized domain expertise and to tailor patient specific requirements – something that is arguably better done post deployment by therapists. The paper reports on a survey conducted among therapists in a rehabilitation clinic aiming to understand end

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 307–312, 2011.

user (therapist) perceptions, background experience with software technology, perceptions about risks and benefits, factors that determine their willingness to apply EUD and perceptions on the organizational support in performing these tasks.

2 Literature Review

In one of the few studies that have taken a broader organizational perspective on EUD, Sutcliffe et al [8] analysed the importance of connecting user motivation to the perceived rewards of using an EUD tool. He concluded that motivation will depend critically on perceived utility and then the actual utility payoff. Mehandjiev et al [5] concluded that participants saw EUD as adding corporate value in terms of better support for agile working practices at an affordable price. However, attitudes to EUD are shaped to a large extent by the culture of the organization and by the benefits. Deci et al. [3] suggests that external intervention via monetary incentives and punishment may undermine intrinsic motivation.

3 Method

A set of open-ended interview sessions with therapists and healthcare researchers were held with the aim of gaining a better understanding of therapists, their experience with technology, work environment, motivation, and creativity skills. Three core themes were identified as critical for enabling EUD practices to take place in the rehabilitation domain:

- Creativity: Do therapists have the creativity required for creating novel technology based solutions and are they motivated to do so?
- Organisation: Support and incentives to nurture a culture of EUD at the workplace.
- Technology: Willingness to work and adopt technology as well as understand software programming concepts i.e. abstract manipulation.

Based on these interviews and a literature survey of related works on organizational perspectives regarding end-user development practice, e.g., see [5], we designed a survey study aiming to evaluate the attitude of therapists regarding these topics.

3.1 Participants

A total of 140 questionnaires were distributed with the support of the management in two rehabilitation clinics; 52 therapists (15 male, 37 female) returned these questionnaires resulting in 37 % overall response rate. Participants were selected based on their knowledge and work with new rehabilitation methods at two research driven rehabilitation clinics in the Netherlands.

The ages of participants ranged from 24 to 62 years with an average of 39.58 years. Twenty one (21) of the participants were occupational therapists; 25, physiotherapists; and 3, speech therapists. The subjects used in this study were identified by their questionnaires as either educated at bachelor level (37%) or have masters degrees (12%).

3.2 Procedure

The questionnaires were distributed in paper form during the summer and autumn of 2010. Subjects were informed that the survey was anonymous, and the statistics gathered would be used for summary purposes only. Participation was voluntary. The questionnaires were followed by a brief interview at a later stage to gather feedback.

3.3 Materials

The questionnaire was designed to investigate therapists' backgrounds as potential End-User Developers, their willingness to create, share, and extend therapy, and their perception of the organisational support available to develop and motivate the creation of new therapies. The final version of the questionnaire consisted of 64 questions. Most questions were presented as a 5-point Likert scale ranging from 1 (strongly agree) to 5 (strongly disagree) – in some cases, 6-point Likert scales were used addressing feedback from pilot runs of the questionnaire.

4 Results

The following summarises the result of the survey.

- Therapists consider themselves *personally creative*. They report being more creative during working hours and when collaborating with others.
- Therapists learn new approaches to therapies mostly from direct interaction with peers rather than from publications or websites, and receive a lot of peer support.
- Therapists are mostly unaware of opportunities to fund new therapy development and rate financial incentive provided as low.
- The result of the survey reveals that therapists are driven by their desire to assist patients rather than to develop technologies. They spend little time on technology development but spend time and effort in developing new therapies.
- Respondents named the following organizational factors that might motivate them to explore new rehabilitation technologies, ranked according to importance: peer recognition (1) more time (2) monetary compensation for extra hours (3) and intellectual ownership of the innovation.
- Therapists are familiar with digital technology but are largely unfamiliar and averse to software programming.
- Therapist attitude is positive towards new technologies if they can improve therapy and are confident that they would bring no risks to patients.

5 Discussion

The results indicate that therapists think that they are slightly more creative when collaborating with other people. We were interested in the possibility of online collaborative applications supporting the dissemination of new therapy approaches supported by technology. Our respondents felt that they understand new things best when learning together with other people in a group setting. The respondents' responses are consistent with the social facilitation theory, which posits that people often perform better in the presence of others than alone [2]. Group learning with peer input is thought to significantly increase learning perceptions, problem solving skills, and help achieve a higher level of learning than individual learning alone [4]. This knowledge allows for the development of competencies and incremental or transformational change and the strong peer support learning.

Regarding technology adoption, respondents felt that they should be given time to learn new tools and attend training sessions. Contribution to training for these users would be funds well invested in keeping those workers motivated by their jobs. It has been cited as a strategy that will increase the likelihood of innovation in end user application [1]. The more positive the perception of organizational support, the greater is the degree of motivation and system utilization [6].

The result of the survey reveals that therapists are driven by their desire to assist patients; adoption of technology thus needs to be related to providing tangible benefits to patients rather than other types of savings of financial benefits. In the Netherlands (and probably also in other countries with similarly organized national health systems), health workers are normally not exposed to extrinsic rewards for creating innovative solutions. Health workers are driven by a high level of moral commitment and intrinsic motivation. The most common form of reward is praise, which can boost an individual's perception of competence and ability. This can inspire a person to continue achieving his goal – thus increasing intrinsic motivation.

The observation of insufficient time as the predominant barrier deserves note. Based on the respondents’ response, it would seem that external incentives such as *more time* to develop new therapies and financial rewards could increase the intrinsic motivation. Awards can be seen as a device that, like intrinsic motivation, motivates individuals to exert an effort. Awards are typically perceived as a gesture of support and are likely to have a positive effect on performance [8].

Some conclusions can be drawn from the data with regards to the feasibility of EUD by therapists: in the current situation therapists are not motivated to become software developers of therapeutic applications. The analysis above identifies factors that could enhance the motivation of therapists to do so (improving treatment, recognition) and remove perceived barriers (allocation of time).

6 Limitation of the Study

This study considers only therapists drawn from two clinics in the Netherlands. Organizational culture and the level of support may vary from country to country and even within. The quantitative nature of these results emphasizes the prevalent attitudes and ignores the possibility that in each organization there will be one or two individuals who take the role of the innovator and who are differentially motivated than the average employee to create, innovate and share their knowledge. To understand how to support these individuals and increase their impact in therapy a more qualitative research approach is needed that will examine motivations, incentives, and barriers these innovators face and how end-user development practices could be enabled through supporting them.

7 Conclusion

Software applications developed by therapists can contribute to major innovations and improvements in health care. The study reported on this paper investigated the ability and motivation of therapists to act as end-user developers, as well as their perception of their organizational support for doing so.

This research examined therapists who have highly specialized expertise and training, but not much exposure or interest in software technology. In contrast to professionals in the creative industries, the therapists 'primary goal is to treat patients, which has to be done within working hours as patients' treatment is confined to working hours. They also do a lot of work outside their regular working hours (e.g. giving courses, workshops etc.). This leaves little room for creativity and therapists are not motivated to create, innovate and share new therapeutic software using new technology; *this is not inherent in their job, their culture or their training*. Therapists were driven by their primary motivation to assist patients in the rehabilitative clinics and the use of IT technology was not considered the primary management tool to achieve their tasks, unlike the subjects of earlier studies employed in industries where IT is central to the business development. Moreover, majority of our respondents did not posses software development skills unlike respondents of previous studies. Perception and attitude towards EUD thus vary according to the job task and the nature of the organization and their computer skills.

The domain of therapy presents some distinct characteristics. This study surprisingly found that although therapists are driven by their moral commitment to help patients, they lack the motivation to explore new opportunities to further their understanding and knowledge regarding software technology. Praise (in its current form) was not sufficient to motivate them to develop new technology-based tools. This finding is in contrast to earlier studies which found that EUDers in other sectors consider peer recognition, training and management recognition of the time spent by EUDers for learning especially outside work hours, sufficient to motivate them to engage or extend the scope of EUD.

This study charts into new territory by providing new insights into the role and significance of gender differences. Male therapists have greater motivation to explore and share new innovative rehabilitative tools than female therapists.

Efforts to introduce EUD in therapy will have implications for the development of interventions and rehabilitation therapy. EUD will encourage therapists to share valuable information necessary in achieving high quality therapy services and optimal patient outcomes. A clinical culture that does not value the importance of implementing EUD and understanding the needs of end user developers will fall short of these goals.

The present study yield valuable insights about the role of rewards and motivation in end-user development. Research to date on EUD has focused on task performance, computer skills, job satisfaction, and technological aspect and implementation barriers of EUD. However, it is difficult to address the issues of growth, barriers and management without in-depth understanding of the EUDers creativity experiences and motivation to address questions arising from implementation of EUD. This study fills this lacuna and provides valuable insights in identifying the EUDers 'attitude towards organizational resources and supportive culture.

This study could be expanded to include a more in-depth study on what motivation techniques would drive the therapists to achieve the goal of new therapeutic technology-based tools and sharing.

Acknowledgement

The authors gratefully acknowledge the support of the Innovation-Oriented Research Programme 'Integral Product Creation and Realization (IOP IPCR)' of the Netherlands Ministry of Economic Affairs, Agriculture and Innovation. The authors thank Dr. H.Seelen and dr. A.Hochstenbach of Adelante Kenniscentrum for their feedback in the questionnaire design and for recruiting respondents at Adelante Rehabilitation Centre; Ingrid van den Tillar, for helping administer the questionnaire at Sint Maartenskliniek; and respondents for their time and effort.

References

1. Cheney, P.H.: Measuring the Success of MIS Development Projects: A Behavioral Approach, Working Paper #1, Iowa State University (1980) (1990)
2. Cook, R.: Social psychology in project management (2001), `http://www.pmforum.org/library/papers/pmpsych1.htm` (retrieved November 29, 2004)
3. Deci, E.L., Koestler, R., Ryan, M.: A meta-analytic review of experiments examining the effects of extrinsic rewards on intrinsic motivation. Psych. Bulletin 125, 627–668 (1999)
4. Hiltz, S.R., Coppola, N., Rotter, N., Turoff, M.: Measuring the importance of collaborative learning for the effectiveness of ALN: A multi-measure, multi-method approach. Journal of A Synchronous Learning Networks 4(2), 103–125 (1999)
5. Mehandjiev, N., Sutcliffe, A., Lee, D.: Organizational View of End-User Development. In: Lieberman, H., et al. (eds.) End User Development, pp. 371–399. Springer, Heidelberg (2006)
6. Jobber, D., Watts, M.: Behavioral Aspects of Marketing Information Systems. Omega 14(1), 69–79 (1986); Sawyer, K.: Group genius: The creative power of collaboration. Basic Books, New York (2007)
7. Schieman, S., Young, M.: The demands of creative work: Implications for stress in the work–family interface. Social Science Research 39(2), 246–259 (2010)
8. Sutcliffe, A., Lee, D., Mehandjiev, N.: Contributions, Costs and Prospects for End-User Development (2003)
9. Tait, R., Walker, D.: Motivating the work force: The Value of External Health and safety awards. Journal of Safety Research 31, 243–251 (2000)

EUD Software Environments in Cultural Heritage: A Prototype

Adalberto L. Simeone and Carmelo Ardito

Università degli Studi di Bari, Dipartimento di Informatica,
via Orabona 4, 70125 Bari, Italy
{simeone,ardito}@di.uniba.it

Abstract. This paper describes the prototype of a framework for designing interactive applications for cultural heritage sites by following an end-user development approach. The framework is devoted to all design stakeholders, i.e. software engineers, HCI experts, cultural heritage experts and visitors, and provides them with tailored design environments for contributing their expertise to shaping the final applications. The design is guided by application templates, which provide the rules for assembling the basic components, called building blocks, whose result is the final application.

Keywords: end-user development, meta-design, cultural heritage.

1 Introduction and Motivation

Since its early ages, information technology has been applied in the cultural heritage domain. Audio guides represented the first type of applications and are still in use in many museums worldwide even today. They provide access to audio descriptions of exhibits present in a museum. In the last decade, various improvements in the used technologies were brought forward. For example, RFID tags and infrared beacons were proposed in order to provide contextual information by identifying the position of the visitor in respect of the museum exhibits. Recent developments have investigated new ways in which technology can be employed to provide more engaging experiences at sites of cultural interest.

Previous work of our research group addressed multimedia educational games to be played at historical sites [1]. In Italy, schoolchildren aged 9-13 are among the frequent visitors of archaeological parks; in order to stimulate their motivation and curiosity, Explore!, a m-learning framework designed to facilitate history learning during visits to sites of cultural interest. This research highlighted three main issues: 1) the need to favor the reuse of existing resources; 2) the inherent difficulties in developing applications for multiple platforms; 3) the advantages in including domain experts and even end users in the design process. In fact, the efforts to create, convert in digital form and maintain the multitude of data related to cultural heritage assets are very high. This is especially true for applications for mobile devices supporting museum or park visitors, as contents have to be adapted to cater to different platforms, screen sizes, input methods, etc. Because of the frequently changing nature of exhibitions, both content and structure of the applications must be updated. This could be

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 313–318, 2011.

carried out by cultural heritage experts if they are empowered with the means to contribute to shaping software artifacts.

An important point of this research concerns the definition of a formal model, the Cultural Heritage Resources model (CHeR), which describes a generic visit experience to a site of cultural interest. It was initially designed to model multimedia resources, in order to facilitate their reuse [2]. However, there are many other types of visitors, who perform various activities at museums and historical sites, exploiting thematic itineraries, exhibitions, interactive installations, etc. In order to reason on all aspects related to different types of visits to sites of cultural interest, the CHeR model encompasses the stakeholders involved in creating the final applications, the digital resources to be presented, the different types of visitors and the relations among all such elements; it better defines the problem space, driving the design of new activities to be performed at cultural heritage sites.

A software framework comprising a set of design environments is in development to allow teams of different experts (software engineers, HCI and domain experts) to cooperate for creating applications according to the CHeR model. A prototype of one of the environments is presented in this paper. In [3] the authors present an EUD environment to assist curators in the design of museum guides. Although similar in concept, our approach, described in Section 2, has a broader aim encompassing all activities related to sites of cultural interest. Our prototype focuses on allowing its users to design such applications in an exclusively visual way by composing building blocks, i.e. elements that represent atomic features of a software application. The activities available in the CHeR are modeled through application templates that expose the available blocks and associated rules. These are described in Section 3; the environment is presented in Section 4. Section 5 concludes the paper.

2 Framework Design

A basic assumption of our approach to the design of interactive systems is that all stakeholders of an interactive system, including end users, are “owners” (or experts) of a part of the problem: software engineers are technology experts, end users are domain experts, Human-Computer Interaction (HCI) researchers are human factors experts, etc. We follow the *Software Shaping Workshop* (SSW) methodology, which prescribes that systems are developed according to a meta-design approach and involving all stakeholders [4]. The software engineers are not any longer the only system developers, but they act as meta-designers: they do not directly create the final systems, but they develop software environments, each targeted to a specific communities of stakeholders, in order to allow them to contribute to system design by bringing their own expertise [5]. The SSW methodology calls such design environments *workshops* to refer to the workshops of some artisans, like carpenters, where every tool suitable for their activity is available.

The communities of stakeholders involved in the design of applications for visiting cultural sites, as addressed by the CHeR model, are: Software Engineers, Human-Computer Interaction (HCI) experts, Cultural Heritage (CH) experts and Visitors. The design occurs in two phases. In the first phase, S*oftware Engineers* use an environment such as Microsoft Visual Studio® to design and develop the workshops for the other communities of stakeholders. They also develop application templates, which provide atomic components and rules to combine them for creating final applications (described in the next section). They collaborate with *HCI experts*, who bring human factors to workshop design. In the second phase, all stakeholders through their workshops contribute to creating the final applications. In particular, *CH experts* contribute to the design primarily by providing proper content and by shaping software artifacts according to the purpose of the final application, e.g. a thematic itinerary in a museum or an educational game in an archaeological park. *Visitors* are the end users, i.e. the persons who will use the developed applications.

3 Application Templates

In software engineering, a template is any processing element that can be combined with a data model and processed by a template engine to produce a result document. In the context of the CHeR model, an *application template* formally is represented by a set of *rules* to assemble together basic elements, called *building blocks*, in order to define the final applications, i.e. those supporting the activities that can be performed or participated at cultural sites, such as: thematic itineraries, interactive installations, educational games, etc. Building blocks are atomic components who expose several functionalities (e.g. showing content, inputting data); they must be completed by inserting different types of multimedia resources (text, image, video, etc). Finally, it defines the devices that the application can be generated to.

Three application templates are available in the current framework, one referring to traditional museum visits (*Museum guide*), one referring to a learning game implemented on mobile phones (*Excursion-game*) [1], and the third one referring to a puzzle game, implemented on a large multitouch screen (*History-Puzzle*) [6]. This latter game is designed to be played by young students at an archaeological park. It proposes puzzles of the 3D reconstruction of historical monuments in the park. In order to solve the puzzles, students have to manipulate with their hands visual widgets shown on the screen representing either puzzle tiles or tiles with questions and answers. In Section 4 we will show how CH experts design a game using the History-Puzzle template. Once a designer (e.g. the CH expert) chooses an application template among those available in his workshop, its associated rules determine the building blocks (B rule) available, the permitted connections among them (C rule) and the multimedia resources that can be inserted in each particular building block (R rule).

A rule of type B can be defined as follows:

$B: AT \underset{allows}{\Longrightarrow} (E_0, E_1, \dots, E_n)$, where:

- AT is an *application template*
- E_i is an element type

In the case of History-Puzzle, rule B is: *"For History-Puzzle application template, the allowed building blocks are of types: Puzzle, Questions and Answers"*.

A rule of type C can be defined as follows:

$C: (c_i, E, p) \underset{allows}{\Longrightarrow} \left(c_j, (E'_0, E'_1, .., E'_n)\right)$, where:

- c_i is a *connection point* of the *left-hand side* (LHS) element *E*
- *p* indicates whether the *connection point* allows 1:1 or 1:N
- c_j is a *connection point* of each one of the *right-hand side* (RHS) elements $E'_0, E'_1, .., E'_n$

An example of rule C is: *"From the source connection point c_i of a building block of type Puzzle (E), it is allowed only one connection (p is of type 1:1) to the target connection point c_j of a building block of type Q&A (in this case the set of elements $E'_0, \dots, E'_n$ is the singleton E'_0)".*

A rule of type R can be defined as follows:

$R: E \Rightarrow (k_0, k_1, \dots, k_n)$, where:

- k_i is a constraint on the E and it is defined as k_i: (min_i, max_i, r_i)
- min_i is the minimum number of occurrences of resource r_i
- max_i is the maximum number of occurrences of resource r_i
- r_i is the type of resource (text, image, video, etc)

An example of rule R is: *"Given a building block of type Puzzle (E), each associated constraint k_i specifies that Puzzle accepts a minimum min_i and a maximum max_i number of resources of type image (r_i)".*

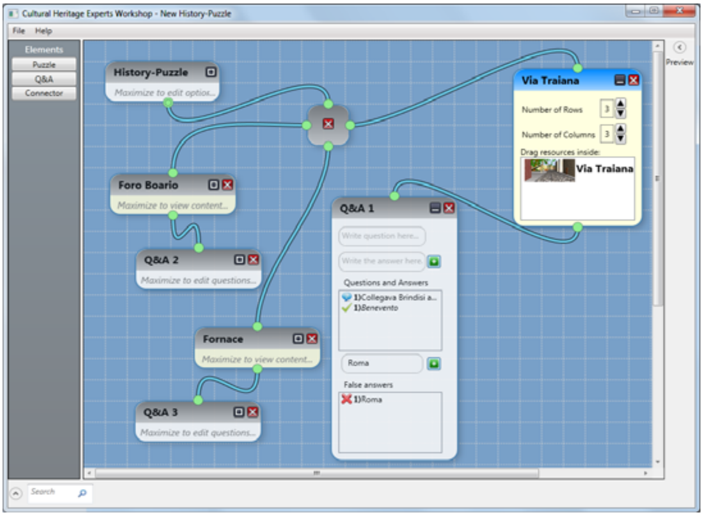
Rules are designed to be extensible by software engineers as they are defined as XML files. In this way, interoperability with elements of future templates is ensured.

4 The Cultural Heritage Experts Workshop

The CH experts workshop offers a visual design environment (inspired by YahooPipes [6]), application templates, building blocks and multimedia resources to allow CH experts to collaborate to the design of applications aimed at visitors of sites of cultural interests. After logging in the workshop, the CH expert (e.g. a male archaeologist) has to choose an application template among those available in his workshop. Application templates, building blocks and multimedia resources are classified according to different visitors' profiles and devices they use.

Let us suppose that the CH expert wishes to design a History-Puzzle game for the archaeological park of Egnathia, in Southern Italy. In his workshop, he selects the History-Puzzle template; an interface, like the one shown in Fig. 1 appears with the *History-Puzzle* block (the root element of the game application) already in place. This element can be maximized by clicking on the button on the top right corner. The root element can be connected to the other elements available for the History-Puzzle template, which are listed in the Elements toolbar on the left side of the interface: Puzzle, Q&A (Questions&Answers) and Connector. The *Puzzle* element is a building block representing a single puzzle of the game, which is usually composed by several puzzles. In order to add a Puzzle, the CH expert drags this element from the toolbar to the main area of the workshop. Then he connects the Puzzle to the root element by drawing a line between the connection points of these blocks. If an element allows 1:N connections, a *Connector* element can be used to multiply that element's connection points. In Fig. 1, three puzzles, Foro Boario, Fornace and Via Traiana have been connected to the root through a Connector.

The CH expert completes each Puzzle by defining the number of tiles in terms of Number of Rows and Number of Columns and by selecting the image of the 3D reconstruction, which will be automatically subdivided in tiles. To include multimedia resources, e.g. the Puzzle image, the CH expert expands the search panel at the bottom of the screen, where resources are classified by type (text, image, audio and video); filter buttons are used to include or exclude the corresponding category from the matches. Then he drags each resource on the accepting element; in the example of Fig. 1, he has added the Via Traiana image in the homonymous Puzzle block. When no resource is present, a watermark label informs users what kind of resources and how many of them the block accepts. When a resource of the wrong kind is dragged over, the cursor will change to indicate that the operation is forbidden.



Number of Rows 3

Number of Columns 3

1)Collegava Brindisi a...
1)Benevento

Roma

1)Roma

Fig. 1. The visual design environment in the CH experts workshop.

The final step is to connect every Puzzle block with a corresponding Q&A building block. First the CH expert drags the Q&A element in the working area and connects it to the associated Puzzle as he has previously done for Puzzles and root. Then he maximizes the Q&A block and types as many questions and matching answers as there are tiles in the connected Puzzle. In order to increment the difficulty of the game, more tiles than necessary can be shown: these additional tiles are created by typing false answers in the corresponding text area of the Q&A block. The preview panel on the right of the interface, when expanded, shows how the content will be presented on the application on the target device with the current presentation template. There are multiple templates for each allowed device in the application template. HCI experts can use their workshop to develop new presentation templates.

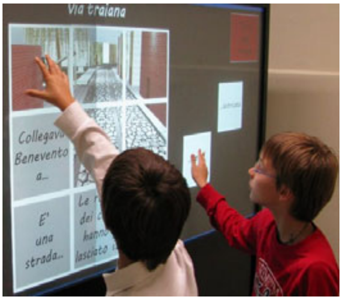


Fig. 2. Two young students playing with a History-Puzzle game

Once the game is ready, the *Export to…* command in the File menu of the CH experts workshop generates a collection of XML documents.

A host application running on the target device uses these files to show the application built in the environment with which visitors can interact. Fig. 2 shows two students interacting on a large multitouch screen with the resulting History-Puzzle developed for the archaeological park of Egnathia. Applications built in this way can be shared and modified once reloaded in the environment.

5 Conclusion

This research work has been motivated by the idea that the design of interactive systems requires a more active participation of all the involved stakeholders. In order to facilitate this process, a framework providing design environments tailored to each community of stakeholders is being developed. In particular, this paper has focused on cultural heritage experts, which play a special role having a deep knowledge both of domain and end users of the applications, namely the visitors of cultural sites. User studies evaluating the impact of the environment in the design process are currently underway and will be reported in a future work.

Acknowledgments. We are grateful to Prof. Maria F. Costabile for her guidance in this research. We also thank the archaeologists of the Department of "Scienze dell'Antichità" of the University of Bari for participating in the design team as CH experts and for testing the CH experts workshop.

References

1. Ardito, C., Buono, P., Costabile, M.F., Lanzilotti, R., Pederson, T., Piccinno, A.: Experiencing the Past through the Senses: An M-Learning Game at Archaeological Parks. IEEE MultiMedia 15(4), 76–81 (2008)
2. Ardito, C., Costabile, M.F., Lanzilotti, R., Simeone, A.L.: Combining multimedia resources for an engaging experience of cultural heritage. In: SAPMIA Workshop at ACM MM 2010, pp. 45–48. ACM, New York (2010)
3. Ghiani, G., Paternò, F., Spano, L.: Cicero Designer: An Environment for End-User Development of Multi-Device Museum Guides. In: Pipek, V., Rosson, M.B., de Ruyter, B., Wulf, V. (eds.) IS-EUD 2009. LNCS, vol. 5435, pp. 265–274. Springer, Heidelberg (2009)
4. Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A.: Visual Interactive Systems for End-User Development: A Model-Based Design Methodology. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 37(6), 1029–1046 (2007)
5. Fischer, G.: Extending Boundaries with Meta-Design and Cultures of Participation. In: NordiCHI 2010, pp. 168–177. ACM, New York (2010)
6. Ardito, C., Costabile, M.F., Lanzilotti, R.: Gameplay on a Multitouch Screen to Foster Learning about Historical Sites. In: AVI 2010, pp. 75–78. ACM, New York (2010)
7. Yahoo Pipes, `http://pipes.yahoo.com/pipes/` (last access on March 14, 2011)

From Consumers to Owners: Using Meta-design Environments to Motivate Changes in Energy Consumption

Holger Dick, Hal Eden, and Gerhard Fischer

University of Colorado at Boulder, ECOT 717, 430 UCB
Boulder, CO 80309, USA
{holger.dick,haleden,gerhard}@colorado.edu

Abstract. Unsustainable energy consumption is a systemic problem facing society that requires technical and social innovations and changes. We argue for understanding and using end-user developments as not just another design principle but as a socio-technical intervention to help people make better decisions as they work to solve such systemic problems. We further explore two established EUD frameworks, Meta-Design and Cultures of Participation, to design systems for one such systemic problem: the energy domain. We present the draft for a system that incorporates principles from these frameworks to inform, motivate, and involve end-users in reducing their energy consumption.

Keywords: meta-design, cultures of participation, energy sustainability, changing human behavior.

1 Introduction

There is overwhelming evidence that our current lifestyle is not sustainable and human energy consumption causes global warming [1]. Governments, industry, and environmental groups are undertaking major efforts to reduce energy consumption, largely resulting in systems that, although technically innovative, are static and closed, viewing the end user as a passive consumer. To reduce energy consumption to sustainable levels, technological innovations and policy changes are not sufficient—changes in human behavior are necessary [2] and systems that involve users as active decision makers [3] are needed.

2 Efforts to Reduce Energy Consumption

Changes in behavior to reduce energy consumption can be fostered through both social and technological interventions. Providing feedback, goal setting, and tailored information are useful in motivating people to change their energy behavior [5]. Steg and Vlek [6] have shown that a *meta-design approach* [7] in which participants are asked to become active in planning their energy environment increases the probability of participants changing their behaviors and saving more energy. Staats, Harland and

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 319–324, 2011.

Wilke [8] found in their longitudinal study that one of the most important contributing factors for changing behaviors and energy savings were supportive social environments. In addition, computer-based feedback mechanisms [9-11] are effective in reducing energy consumption [2, 5, 12] and have been implemented and analyzed in the HCI domain.

3 Motivating People with Socio-technical Environments

To reach the goal of reducing energy consumption at a societal level, socio-technical interventions [13] that go beyond simple presentations of facts are necessary; they need to make use of new insights into social and behavioral psychology to *motivate* consumers. We have identified two mechanisms, *psychological ownership* and *motivating social environments* to involve consumers that could be facilitated by software systems but are currently being ignored in the energy domain.

Psychological ownership [14] describes a state in which a person feels closely connected to an object or idea, to the degree that it becomes part of an 'extended self'. As soon as people see something as their own, they value it higher and are more likely to invest time and effort in it.

In a meta-review of research on psychological ownership, Pierce and colleagues have found several requirements for psychological ownership: (1) *control*, (2) *investment of self*, (3) *intimate knowing*, and (4) *modifiable targets* [14]. If an object or an idea fulfills all of these requirements, people are more likely to feel ownership for this target.

While the technological foundations for these requirements are currently created with *smart grids (http://www.oe.energy.gov/smartgrid.htm)*, the software infrastructure available to end-users does not make use of them. In almost all developments of smart grids to date, consumers are given very limited *control*. The technical implementation and the utility companies do *not reward investment of self*. The grid is designed as a system for passive consumers in which consumers are given the same monthly bills that list overall consumption in the abstract unit kwH, *not supporting intimate knowing* of how energy is being used or how energy could be saved. Finally, the only thing that consumers can change in the current smart grid is which devices they use and how often they use them; the system does *not provide any means of modifiability* to end-users.

Motivating Social Environments. Although changes in the social environment have been shown to cause people to use less energy [15, 16], supportive social environments are not commonly used to reduce energy. Unfortunately, current energy infrastructures prevent consumers from creating social norms or peer pressure.

Social proof [17] describes the effect that people act a certain way because they observe others acting this way. In such situations, the fact that others chose something acts as *proof* that this choice is preferable. However, energy consumption is completely individualistic and invisible to the consumers themselves and to others [2]. Aside from choosing to drive a Toyota Prius as a symbol of energy-efficiency or installing solar cells to show support for renewable energies, people have few way to share their energy attitudes or behaviors. Thus, for highly energy relevant behaviors like the temperature of the thermostat, the installation of house insulation, or the choice of appliances, no generally established social norms exist that could motivate and guide consumers to reduce energy consumption. Without awareness of other people's actions, no social proof can be created.

4 Conceptual Frameworks

To address and implement the two ways of motivating people mentioned above, we have found two conceptual frameworks to be particularly helpful, namely *meta-design* and *cultures of participation*. They are well suited for systems that motivate and involve end-users and thus offer themselves to the design of energy-relevant systems.

Meta-Design. Meta-design environments [7] are solution spaces [18] in which users are able to identify, explore, and reassess their needs during use time and act as designers that can change the environment accordingly when needed.

An important element of a meta-design environment is the *"Seed-Evolutionary Growth-Reseed"* model [7]. In this model, designers do not attempt to build a complete system; instead they create *seeds* for users that provide basic functionality and can be modified by end-users. All users can modify and expand the seed in the *evolutionary growth* phase before the designer *reseed* the system with the contributions made by the community.

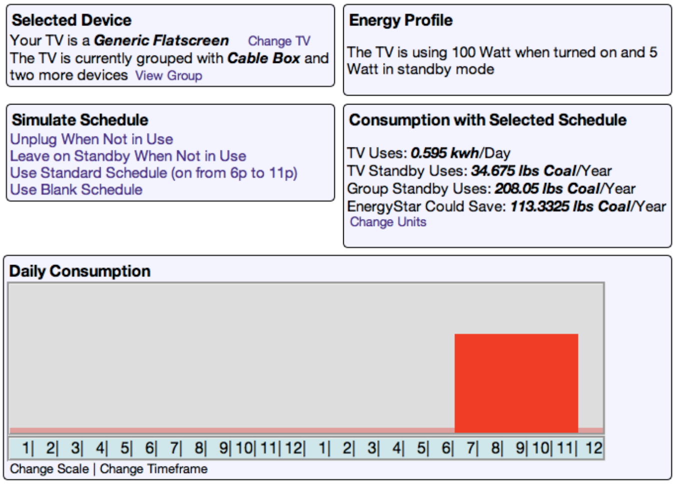
Meta-design environments foster psychological ownership by giving users control and openness and rewarding investment of self in the ongoing development of the system. Being an owner of the system makes people more likely to prefer the system to others, invest more time in it, and develop extensions to it. Their own extensions, in return are something for which users are likely to feel responsible for, increasing their feeling of ownership and their motivation to contribute on an ongoing basis.

Cultures of Participation [19] offer a new platform for human connection, bringing together otherwise unconnected individuals and replacing common background or geographic proximity with a sense of well-defined purpose and the successful common pursuit of this purpose as the condensation point for human connection.

Cultures of participation and meta-design environments are tightly integrated [20]. To be a successful meta-design system, users have to be able to share their ideas and developments, to get help from other users, and to find extensions and developments that have already been implemented by others; they have to form a culture of participation. Cultures of participation require the underlying software system to be open and modifiable so that users can participate in meaningful problems. The software underlying a culture of participation has to be dynamic and has to allow users to adapt it to their needs, and to reseed their own developments with others in the community; meta-design environments are needed. Cultures of participation are well suited to foster and support motivating social environments in which people can create social proofs and social norms by providing tools for sharing and for creating awareness. Since people are not merely consumers of a system but active participants of a community, they are more likely to be influenced by the actions and opinions of others.

5 EMPIRE–A System to Reduce Energy Consumption

We are currently building EMPIRE (EMPIRE = Empower People in Reducing Energy Consumption), a meta-design environment in which users can measure, simulate, and visualize their energy consumption. We are following a design approach with the two conceptual frameworks providing the overall design and functionality of the system. In the design of the prototypes, we use personas that are based on a crowd-sourced survey using Amazon Mechanical Turk. This approach allows us to cover a wide variety of potential users, interest, and preferences; for the final versions, the personas will be based on the actual users of the system to more accurately fit their specific needs.



Selected Device

Your TV is a ***Generic Flatscreen*** Change TV

The TV is currently grouped with ***Cable Box*** and two more devices View Group

Energy Profile

The TV is using 100 Watt when turned on and 5 Watt in standby mode

Simulate Schedule

Unplug When Not in Use
Leave on Standby When Not in Use
Use Standard Schedule (on from 6p to 11p)
Use Blank Schedule

Consumption with Selected Schedule

TV Uses: ***0.595 kwh***/Day
TV Standby Uses: ***34.675 lbs Coal***/Year
Group Standby Uses: ***208.05 lbs Coal***/Year
EnergyStar Could Save: ***113.3325 lbs Coal***/Year
Change Units

Daily Consumption

Change Scale | Change Timeframe

Fig. 1. First Prototype of EMPIRE

The first iteration of EMPIRE aimed to address the problem of a missing psychological ownership. Using a meta-design approach, we created several prototypes that let users explore and visualize their own energy consumption by answering questions about their energy profile and integrating a simple energy simulator (see Figure 1). The goal of this approach was to foster *intimate knowing* by providing the ability to explore the causes and effects of consumption in detail. Initial informal tests with users showed that people were surprised by the results and expressed the opinion that prototypes gave them insights that had not occurred to them before. One participant found, to his surprise, that using a power-strip to turn off his cable box and DVD player at night would save him more energy than getting a new Energy Star certified TV–saving the money he would have spent on a new TV.

We are currently evaluating our early prototypes in crowd-sourced user studies, using Amazon Mechanical Turk [21]. In these studies, we measure how the different systems and different representations influence the users' decision-making processes and opinions about their energy consumption.

The next steps will be to further improve the meta-design aspects of the system and let all users *modify* and expand the system to their needs by creating simulations and visualizations that are meaningful to them. Currently, they can combine and select elements but not edit or create new ones. These steps should offer further reward for the *investment of self* and offer more *control.*

Finally, the individual's actions will be integrated with a culture of participation, so that users can share their creations and insights, help others, gain social recognition, and take on leadership roles within the community, thereby fostering and *rewarding the investment of self.* We will use our experiences and insights from our former work on supporting Cultures of Participation [22] and implement awareness tools [23], that will allow the community of EMPIRE participants to share and become aware of people's energy improvements, their insights, their behaviors, and their consumption. These tools build the foundation for a *supportive social environment* in which energy usage becomes social.

6 Conclusions

Meta-design and cultures of participation are promising frameworks for the development of more-involving and motivating energy systems. There is ample support in the literature that helping people to become psychological owners of their personal energy domain and to become part of a supportive social environment are effective and underused ways to reduce energy consumption. Our system building efforts are at an early stage and more user testing is needed to evaluate the effectiveness of our own implementation of the conceptual frameworks.

Acknowledgments. The research is supported in part by two grants from the National Science Foundation: (1) IIS 0968588 "SoCS: Energy Sustainability and Smart Grids: Fostering and Supporting Cultures of Participation in the Energy Landscape of the Future" and (2) OCI 1028017 "CDI-Type I: Transformative Models of Learning and Discovery in Cultures of Participation".

References

1. Intergovernmental Panel on Climate, C.: Climate Change 2007: Synthesis Report. Summary for Policymakers. Intergovernmental Panel on Climate Change (2007)
2. Ehrhardt-Martinez, K., Donnelly, K.A., Laitner, J.A.S.: Advanced Metering Initiatives and Residential Feedback Programs: A Meta-Review for Household Electricity-Saving Opportunities, vol. E105. American Council for an Energy-Efficient Economy, Washington, D.C, p. 128 (2010)
3. Fischer, G.: Beyond 'Couch Potatoes': From Consumers to Designers and Active Contributors. Firstmonday (Peer-Reviewed Journal on the Internet) (2002)
4. Consolvo, S., McDonald, D., Landay, J.A.: Theory-Drive Design Strategies for Technologies That Support Behavior Change in Everyday Life. In: Proceedings of Chi 2009, Boston, pp. 405–414. ACM, New York (2009)

5. Abrahamse, W., Steg, L., Vlek, C., Rothengatter, T.: The Effect of Tailored Information, Goal Setting, and Tailored Feedback on Household Energy Use, Energy-Related Behaviors, and Behavioral Antecedents. Journal of Environmental Psychology 27, 265–276 (2007)
6. Steg, L., Vlek, C.: Encouraging Pro-Environmental Behaviour: An Integrative Review and Research Agenda. Journal of Environmental Psychology 29, 309–317 (2009)
7. Fischer, G., Nakakoji, K., Ye, Y.: Meta-Design: Guidelines for Supporting Domain Experts in Software Development. IEEE Software, 37–44 (September/October 2009)
8. Staats, H., Harland, P., Wilke, H.A.: Effecting Durable Change. A Team Approach to Improve Environmental Behavior in the Household. Environment and Behavior 36, 341–367 (2004)
9. Holmes, T.: Eco-Visualization: Combining Art and Technology to Reduce Energy Consumption. In: Proceedings of Creativity & Cognition, pp. 153–162. ACM, Washington, DC (2007)
10. Froehlich, J., Findlater, L., Landay, J.: The Design of Eco-Feedback Technology. In: Proceedings of CHI Conference, pp. 1999–2008. ACM, Atlanta (2010)
11. Kirman, B., Linehan, C., Lawson, S., Foster, D.: There's a Monster in My Kitchen: Using Aversive Feedback to Motivate Behaviour Change. In: CHI 2010, pp. 2685–2694 (2010)
12. Fischer, C.: Feedback on Household Electricity Consumption: A Tool for Saving Energy? Energy Efficiency (2008)
13. Mumford, E.: A Socio-Technical Approach to Systems Design. Requirements Engineering 5, 59–77 (2000)
14. Pierce, J.L., Kostova, T., Dirks, K.T.: The State of Psychological Ownership: Integrating and Extending a Century of Research. Review of General Psychology (2002)
15. Schultz, P.W., Nolan, J.M., Cialdini, R.B., Goldstein, N.J., Griskevicius, V.: The Constructive, Destructive, and Reconstructive Power of Social Norms. Psychological Science: a Journal of the American Psychological Society / APS 18, 429–434 (2007)
16. Schultz, W., Khazian, A., Zaleski, A.: Using Normative Social Influence to Promote Conservation among Hotel Guests. Social Influence 3, 4–23 (2008)
17. Cialdini, R.: Influence: Science and Practice, 5th edn. Pearson, Boston (2009)
18. Giaccardi, E., Fischer, G.: Creativity and Evolution: A Metadesign Perspective. Digital Creativity 19, 19–32 (2008)
19. Fischer, G.: Cultures of Participation and Social Computing: Rethinking and Reinventing Learning and Education. In: Proceedings of the International Conference on Advanced Learning Technologies (ICALT), pp. 1–5. IEEE Press, Riga (2009)
20. Fischer, G.: End-User Development and Meta-Design: Foundations for Cultures of Participation. In: Pipek, V., Rossen, M.B., deRuyter, B., Wulf, V. (eds.) End-User Development, pp. 3–14. Springer, Heidelberg (2009)
21. Kittur, A., Chi, E.H., Suh, B.: Crowdsourcing User Studies with Mechanical Turk. In: CHI 2008, Florence, Italy. ACM, New York (2008)
22. Dick, H., Eden, H., Fischer, G.: Increasing and Sustaining Participation to Support and Foster Social Creativity. In: Proceedings of the International Conference on Creativity and Cognition (C&C 2009), Berkeley, Ca, pp. 363–364 (October 2009)
23. Kimmerle, J., Cress, U., Hesse, F.W.: An Interactional Perspective on Group Awareness: Alleviating the Information-Exchange Dilemma (for Everybody?). International Journal of Human-Computer Studies 65, 899–910 (2007)

Collective Programming: Making End-User Programming (More) Social

Alexander Repenning[1], Navid Ahmadi[2], Nadia Repenning[1], Andri Ioannidou[1], David Webb[3], and Krista Marshall[3]

[1] AgentSheets Inc., Boulder, Colorado
[2] University of Lugano, Switzerland
[3] University of Colorado, Boulder, Colorado
{alexander,nadia,andri}@agentsheets.com, ahmadin@usi.ch,
{dcwebb,krista.marshall}@colorado.edu

Abstract. The do-it-yourself Web 2.0 culture is quickly creating and sharing more end-user produced content. Gradually moving from static content, such as pictures and text, to interactive content, such as end-user programmed games, the artifacts created and shared have become significantly more sophisticated. The next frontier to make end-user programming more social is to move beyond the current create, upload, share, download, and repeat Web 2.0 models. Collective Programming is a framework that fuses 100% Web-native end-user programming tools with real-time communication mechanisms into a cloud-based multi end-user programming environment. A prototype built, called CyberCollage, enables groups of students to work on game design projects together: they can play multi-user games, change game worlds in real-time, and engage in virtual pair programming.

Keywords: collective programming, end-user pair programming, computers and education.

1 Introduction

The 21st century do-it-yourself Web 2.0 culture is quickly creating more end-user produced content. From sharing static content such as pictures (e.g. Flickr), encyclopedic articles (e.g., Wikipedia) and dynamic content such as movies (e.g. YouTube), end-users are gradually progressing to *interactive* content such as end-user modded [1] games (e.g. LittleBigPlanet) and end-user created programs (e.g. Yahoo pipes). At the same time, improved infrastructure including faster networks, ubiquitous internet connectivity, audio and video capabilities built into basic computers enables *real-time* communication of potentially large numbers of end-users to create sophisticated computational artifacts such as games and simulations. Already, the combination of tools such as screen sharing, chat, voice over IP, and shared white boards is facilitating new kinds of collaboration including training, design and research at great distances.

CyberCollage is a first-of-a-kind real-time end-user development environment for creating interactive content. Implemented as a cloud-based Web application,

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 325–330, 2011.

CyberCollage enables a new form of social end-user development that we call *Collective Programming*. At the content level, CyberCollage allows end users to build sophisticated computation artifacts such as games and simulations using drag and drop visual programming approaches developed previously in AgentSheets [2], AgentCubes [3] and numerous other educational programming environments including Alice [4], Scratch [5], and Squeak Etoys [6]. An important contribution of CyberCollage over these previous systems is that the entire programming and runtime environment is 100% browser based and built with Web-native HTML 5 technologies. Even more important for Collective Programming, however, is the real-time communication framework built into CyberCollage allowing groups of end-user programmers to collaborate concurrently on projects shared in the cloud. Imagine, for instance, two end users collaborating on a Frogger-like video game (Fig. 1). They can simultaneously play the game, e.g., have a two-frog race, change the game world, or modify the game behavior.

The real-time communication aspect of Collective Programming is essential for enabling a new kind of social end-user programming that we believe to be especially useful in educational settings. The challenge is not just to send more information faster, but also to maintain the perception of simultaneity among a group of collaborators. For instance, if Tim and Victoria are using or modifying their game (Fig. 1) their perception of the interaction should be as though they were sitting next to each other in the physical world. Significant motivational and peer learning [7] benefits of pair programming [8] have been documented. CyberCollage could be considered a virtual pair-programming environment that enables end users to program together remotely.

2 CyberCollage: Real-Time Collaborative End-User Programming

CyberCollage enables real-time collaboration through a combination of formal and informal communication. At the formal level, participants share a common artifact (game or simulation) consisting of media (images), programs (agent behaviors), and game/simulation worlds (worksheets) and communicate through the exchange of the modified pieces of the artifact. Informally, participants communicate through online communication mechanisms such as chat, voice or video. They also communicate through awareness interfaces that keep them informed about other participant actions.

CyberCollage significantly advances the state of real-time social interfaces in ways that are probably best explained through a scenario. Imagine that two students, Tim and Victoria, are working on a joint Frogger-like game (Fig. 1). Tim wants to work on the frog whereas Victoria is eager to build the road and the truck. They start a new project and work on their respective game objects in real time. Each person has a *focus* defined by what they are working on and a *peripheral vision* providing social information about what others are doing. Even with Victoria's focus on programming the truck, a *presence (or awareness) interface* allows her to perceive that Tim is drawing the Frog. This presence information may trigger the need, or opportunity, to engage in additional communication. They will not only be able to see how the other person doing, but will also be able to take control and contribute in real time. That is, Victoria can touch up Tim's frog image and Tim can help Victoria program the truck.

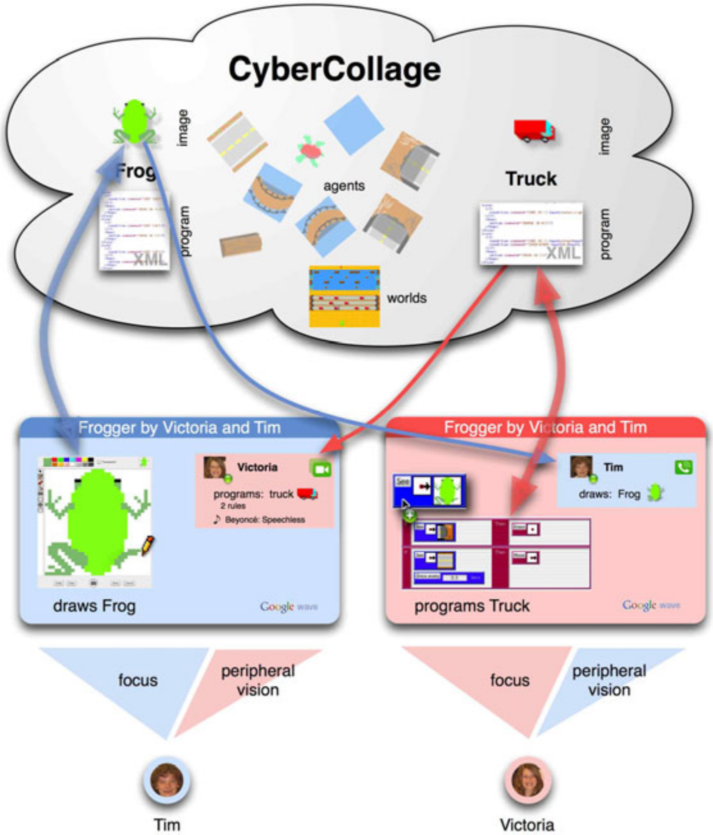


Fig. 1. Collective Programming = Real-Time Communication + Rich Interactive Content + Web-based End-User Programming. Tim and Victoria collectively author a Frogger game. Tim draws the frog, Victoria programs the truck. Through peripheral vision implemented as presence interface they can track what the other is doing and interact in real time.

Technically speaking, CyberCollage is a cloud-based integrated development environment (IDE) with client and server side components. On the front-end, Javascript clients provide end users with an HTML 5-based visual programming environment, operating inside a Web browser. The IDE lets users create agents, draw depictions, program the agent behavior in a visual programming environment, and execute the agents in the worksheet. On the back-end, a communication component synchronizes multiple clients working on the same project. While users interact with their programming environment individually, the client sends updates to the server. The updates include changes in agent depictions, program updates, and the worksheet modifications. The server collects the updates from each client and broadcasts them to all other clients. Each client has a dispatching component that fetches the updates from the server and dispatches them to the IDE.

3 Related Work

Fig. 2 depicts a two dimensional space of collaboration models. Horizontally, the Use⇔Design continuum captures a number of points, including just using a finished artifact, changing it (modding), and programming one from scratch. Vertically, an essential distinction is made between real-time, synchronous collaboration and off-line, asynchronous interaction.

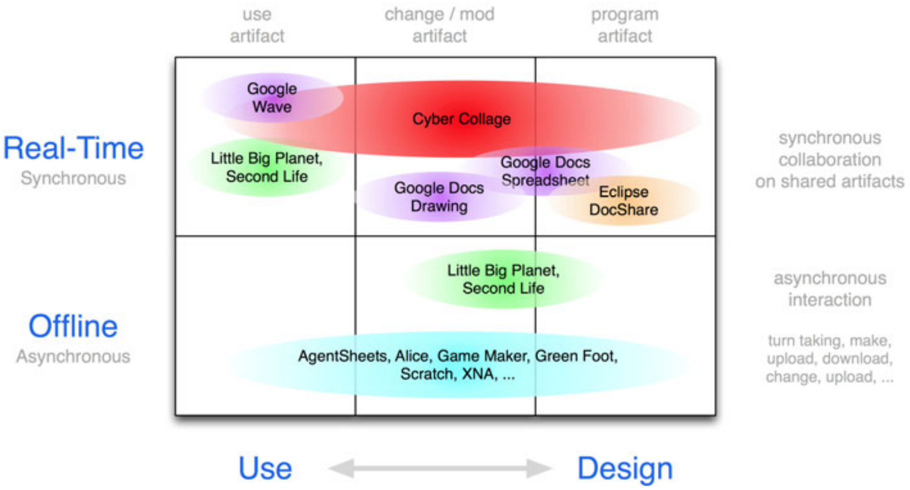

use artifact
change / mod artifact
program artifact
Real-Time
Synchronous
Offline
Asynchronous
Google Wave
Cyber Collage
Little Big Planet, Second Life
Google Docs Spreadsheet
Google Docs Drawing
Eclipse DocShare
Little Big Planet, Second Life
AgentSheets, Alice, Game Maker, Green Foot, Scratch, XNA, ...
synchronous collaboration on shared artifacts
asynchronous interaction
turn taking, make, upload, download, change, upload, ...
Use
Design

Fig. 2. Collaboration Models

Offline collaboration models are based on the asynchronous interactions of participants to use, mod, or design games and simulations. A typical interaction at this level may start with one person making a game, uploading it to a repository and then having a different person play the game. This, in turn, may interest the second person to change the game world or even to change the game behavior. This would typically require downloading the game, assuming its source is available, modifying it, and then perhaps uploading the new version back to the repository.

The synchronous nature of real-time collaboration is radically different from the asynchronous type of offline interaction. In this model people are working together on shared artifacts. The real-time character of this approach requires more sophisticated client/server architectures providing collaborators a sense of presence. Who is online? Who is working on what and how? If one person is changing something then this change needs to be broadcasted to all other users as fast as possible. The enormous degree of interactivity may require access control, e.g., file locking, and interaction protocols such as turn taking to avoid potential conflict. Artifacts are shared.

CyberCollage is a real-time collaboration framework covering the entire Use⇔ Design continuum. With it end users can use games and simulations, change them, and program new ones collectively. CyberCollage allows, theoretically, any number of users to participate in all aspects of the Use⇔Design continuum. Conceptually speaking, CyberCollage combines and extends ideas found in frameworks such as Google Wave, Google Docs Drawing and Google Docs Spreadsheet. Combining the ideas from Google Wave and Google Docs Drawing two users could collaboratively build a chess game by creating all the pieces, drawing a chess board, placing the pieces onto the board and moving them on the board taking turns. However, the ability to program the game would allow them to quickly advance from the collaborative drawing to implementing multi-user games or simulations.

4 Evaluation

To evaluate CyberCollage as an environment for real-time collaborative design, education researchers from the University of Colorado conducted a pilot study. Three sessions involving middle grades students with prior experience with AgentSheets were organized to document student interaction and assess end-user feasibility.

In one activity, two boys started with a highly structured, competitive programming activity. However, they soon expressed that they did not like to compete but collaborate. Their interaction organically evolved into a side-by-side game design session. The pair immediately began adding, drawing, and programming agents, creating a sophisticated two-player 'good versus evil' game. The students became so engaged in their game creation that they continued working on it after class and at home. Once their game was functional, the communication between the pair shifted from collaborative design to negotiation of agent behaviors. Communication occurred through verbal interaction as well as through observing agent behavior in the programming environment and in successive rounds of game play. For example, when one student observed that his partner's agent was immune to his "laser beam", he adapted his agent's behavior to gain an advantage. Eventually, the collaborative design session morphed again into a competitive programming arms race in which two boys pitted strategic programming methods against each other and, at times, renegotiated the rules of the game.

Another activity involved five students – two from Colorado and three from Wyoming – communicating over a Skype video connection while using CyberCollage. Students were encouraged to design a Frogger-like game. With five students involved, the group required leadership and some negotiation of roles [9]. Over a period of 90 minutes we observed an emergent participatory structure in which students communicated verbally, as needed, but found ample information available on their collective workspace as new agents were designed, positioned, and programmed by group members [10]. Tasks were initially assigned by one of the girls from Wyoming, but over time all students provided input on additions, deletions and adaptations of the game objects.

While space limitations preclude a more detailed summary of the analysis of student participation and collaboration, the initial pilot of CyberCollage shows promise as an environment for real-time design for both local and distant collaborators. Real-time, synchronous design and programming were accessible and engaging for middle grades students, demonstrating that they could collaboratively develop a working game in one to two hours.

Acknowledgements

This work was supported by the National Science Foundation (grant number IIP-1014249). Opinions expressed are those of the authors and not necessarily those of the National Science Foundation.

References

[1] El-Nasr, M.S., Smith, B.K.: Learning through game modding. Computers in Entertainment 4(1), Article 7 (January 2006)

[2] Repenning, A., Ambach, J.: Tactile Programming: A Unified Manipulation Paradigm Supporting Program Comprehension, Composition and Sharing. In: Proceedings of the 1996 IEEE Symposium of Visual Languages, Boulder, CO, pp. 102–109 (1996)

[3] Ioannidou, A., Repenning, A., Webb, D.: AgentCubes: Incremental 3D End-User Development. Journal of Visual Languages and Computing, Special Issue on Best Papers from VL/HCC 2008 20(4), 236–251 (2009)

[4] Moskal, B., Lurie, D., Cooper, S.: Evaluating the effectiveness of a new instructional approach. In: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, ACM, Norfolk (2004)

[5] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y.: Scratch: Programming for All. Communications of the ACM 52(11) (2009)

[6] Squeakland, Home of Squeak Etoys, http://www.squeakland.org/

[7] Wills, C.E., Deremer, D., McCauley, R.A., Null, L.: Studying the use of peer learning in the introductory computer science curriculum. Computer Science Education 9, 71–88 (1999)

[8] Williams, et al.: Strengthening the case for pair programming. IEEE Software 17(4), 19–25 (2000)

[9] Fuchs, L.S., Fuchs, D., Kazdan, S., Karns, K., Calhoon, M.B., Hamlett, C.L., Hewlett, S.: Effects of workgroup structure and size on student productivity during collaborative work on complex tasks. Elementary School Journal 100(3), 183–212 (2000)

[10] De Laat, M., Lally, V., Lipponen, L., Simons, R.-J.: Investigating patterns of interaction in networked learning and computer-supported collaborative learning: A role for Social Network Analysis. Computer-Supported Collaborative Learning 2, 87–103 (2007)

Using Recommendations to Help Novices to Reuse Design Knowledge

Paloma Díaz, Alessio Malizia, Ignacio Navarro, and Ignacio Aedo

DEI Lab, University Carlos III de Madrid
Avenidad de la universidad 30, 28911, Leganés, Spain
{pdp,amalizia,inmartin}@inf.uc3m.es
aedo@ia.uc3m.es

Abstract. The use of pattern languages is not so straightforward since itsusers have to identify the patterns they need, browsing the language and understanding both the benefits and trade-offs of each pattern as well as the relations and interactions it has with other patterns. Novice designers might benefit from tools that assist them in this learning task. In this paper we describe a recommendation tool embedded in a visual environment for pattern-based design which aims at suggesting patterns to help novice designers to produce better designs and understand the language.

Keywords: Collaborative filtering,Design Patterns, end-user development.

1 Introduction

Design patterns are supposed to help novices, including casual designers and end-users,to produce better designs, since each design pattern encapsulates a piece of design knowledge based on real experience.Patterns are usually organized as a cohesive language [6]so when novice designersapproach complex design problems that involve the application of several patterns, they can navigate through the language to identify successful design options. However, the use of pattern languages is not as straightforward. The chance to identify the right patterns strongly depends on the designers experience and on the communicability of both the patterns and the relationships used in the language. Taking into account that pattern languages should support novices we have to look for ways to assist them in finding the solutions they need.In this context there are two ways to try to enhance the pattern searching process. On the one hand, we can try to reduce the cognitive gap between the designer perspective of the problem and the pattern description byusing notations more meaningful for the user.On the other, we can help novices to understand how patterns relate by making explicit framework, that in the sense of [2] arecollections ofpatterns thatexperienced designers usually apply together. In this paper we propose a recommendation system that has been built upona visual tool that follows the first approach (reducing the cognitive gap) to deal with the second one (knowing how experts do combine patterns). We will describe the tool and a limited experiment with novice designers.

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 331–336, 2011.

2 Related Works

In an empirical study [4] it was found out that novices apply different strategies to browse pattern languages and most of them are driven by the need to understand whether a design pattern contributes or not to solve a specific design goal. VEISIG is a tool aimed at fulfilling this browsing strategy [3]. It represents a pattern language using a visually enhanced and interactive graph that puts the stress on design goals, since theyseem to be closer to the designers' perspective of the problem. The tool also highlights therelations amongst patterns so that designers can realize how they interact. This way of interacting with the language was considered useful in an experiment with non-expert designers [3] but participants didn't identify all the patterns required to solve the problem they were proposed. To deal with this flaw,the concept of framework [2] can be implemented as a recommendation system. Recommendation systems rate collections of items taking into account several sources of knowledge. To the best of our knowledge there is only one work about recommendation anddesign patterns and [1]but it is oriented towards identifying the patterns required for a given problem description. Our goal is to help casual designers to reuse the design knowledge underlying patterns languages that can be inferred from the way more experience designers use them; but still we aim at letting them the initiative to explore the language and take the decisions about the patterns they need to use.

3 The Recommendation Module

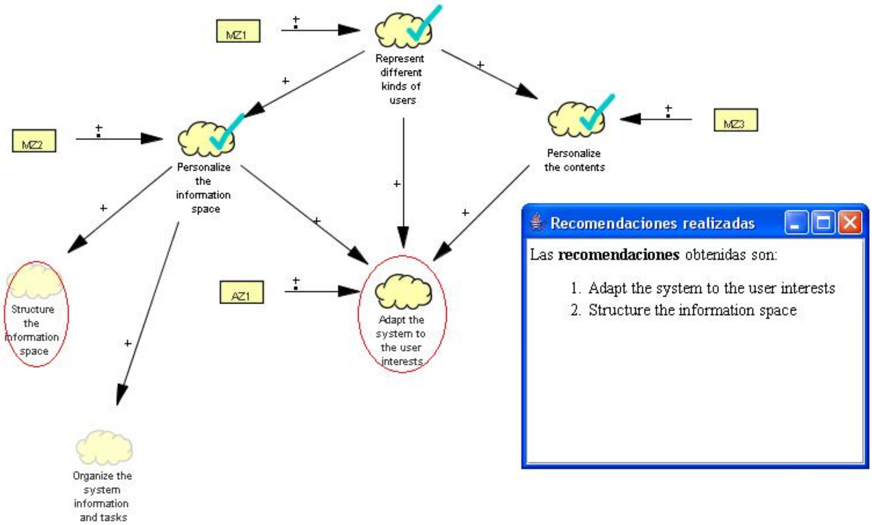
In this paper we propose the creation of a recommendation module for VEISIG in order to help novice designersto improve theirdesigns by reusing design patterns and the knowledge of more experienced designers.The recommendation module is based on collaborative filtering to obtain a rating for each pattern of the pattern space.According to literature there are four kinds of recommendation systems [1]: content-based; collaborative filtering; hybrid systems and preference-based filtering systems. Among them only collaborative filtering do not rely on the previous experience of the same user; which is our case would be useless. Instead of that it rates the items to recommend using the ratings that other users gave them. In our case, the knowledge base is made up of the pattern language (whose items and structure gather explicit expert design knowledge) and the solutions made by expert users, that is, implicit expert design knowledge.

VEISIG provides users witha visual representation of the pattern language using design goals. Goals are related through structural relations (AND and OR), positive contributions or negative relations (Hurt and Break). Patterns, which are the boxes in Fig. 1, are tied to the goals they satisfy which are the clouds in the figure. Goals are organized in six design views to facilitate users' exploration: structure, navigation, presentation, personalization and security. Once the user has selected the goals (those with a tick in fig. 1), she can ask for recommendations. The recommendation algorithm is described below.

First, it rates allthe patterns thatare not in the initial user selection (MZ1, MZ2 and MZ3 in the example). The rating of a pattern indicates its affinity with the initial set and it is obtained using three factors:

a) The number of occurrences where the pattern appears in combination with any of the patterns in the initial set.
b) The types of relation with the patterns in the initial set. In the example, AZ1 is connected with all the patterns in the initial set with a contribution relation, so it rating is increased.
c) If a pattern is categorized in the same design viewthan patterns in the initial set, its rating is increased.

Once all the ratings are obtained, the recommendation algorithm chooses the pattern with higher rating. From that pattern, the results are filtered to produce a number of manageable recommendations, that shouldn't be higher than the number of patterns in the initial set. Recommendations never include patterns that have a conflict relationship with those in the initial set or with patterns with a higher rating in the recommendation set. Recommended goals (patterns) are highlighted in the visual representation and shown in a popup window.


MZ1
+
Represent different kinds of users
+
+
MZ2
+
Personalize the information space
+
Personalize the contents
+
MZ3
+
+
+
Structure the information space
+
AZ1
+
Adapt the system to the user interests
Organize the system information and tasks

Recomendaciones realizadas

Las **recomendaciones** obtenidas son:

1. Adapt the system to the user interests
2. Structure the information space

Fig. 1. Example of recommendation

In the image the goals initially selected by the user are those with a tick. Once the recommendation module is fired, two feedbacks are provided: a popup window shows the goals recommended and they are also highlighted in the visual representation using a circle. Users then, may choose some of the patterns recommended to extend their solution or they can skip the recommendation and continue with the design process selecting other patterns.

4 Evaluation

We made a preliminary evaluation study of the recommendation module trying to achieve two goals: (1) check if the recommendation system helps to select patterns to improve a design, and (2) if it is useful for novice designers.Theparticipants (evaluators so forth)are undergraduate students from computer engineering who didn't know anything about the patterns language. They can be considered novice designers as their knowledge on web design is more concerned with technical issues than with usability ones, which are those gathered in the pattern language. 39 students took part in this study.

First of all we gave them a brief introduction onthe tool functionalities. Then, the evaluation was divided in three phases. In the first one, they were prompted with a group of tasks that could be solved using VEISIG and the recommendation module. Alltaskshad the same structure: a problem with an incomplete solution. We asked them to improve the solutions with new patterns. The second phase was a questionnaire used to collect users' perceptions and opinions about the system as well as suggestions to improve it. Table 1 summarizes the questions included in the form.

Table 1. Questions about VEISG and its recommendation module

Q_1	Does the recommendation module help in the navigation of web pattern space?
Q_2	Were your strategies improved by the recommendation system?
Q_3	What should be added to improve the navigation quality?
Q_4	Are you satisfied with the proposed solutions?
Q_5	How would you improve the quality of the solutions?
Q_6	In your opinion is this component useful for expert users or for novice users (or both)? Explain your decision.
Q_7	How useful was the system to understand the patterns and their relationships?

Q_3, Q_4, Q_6, Q_7were open, whilstQ_1, Q_2 and Q_5 used a 5-values like scale.The first three questions were about using the language (i.e. browsing the design space). Q4 and Q5 collect information about the usefulness of the system for designing solutions, and Q6 gather their impression on whether the system was useful for novice users or not. The last question, Q7, was focused on getting feedback about whether the module could help to understand the patterns language. Finally they had to measure their expertise in design patterns, web design and other design issues. As expected, most of the users were knowledgeable in web design and design patterns in general, though they did not know anything about VEISIG and its pattern language.So our participants could be considered as novice users for this language: they didn't know the patterns in advance or the way they were organized in the language but they were able to design solutions.

The main findings we got from the answers to these questions are the following:

- *The recommendation algorithm is considered useful.* Mostparticipants said that they felt the design time had been reduced and helped to obtain better solutions. Indeed all of them improved the initial solutions they were proposed when they used VEISIG to look for more patterns. A great number of the participant use the recommendation made by the system to improve their solutions.

- *Novice users might find the recommendation tool useful not only to improve their designs but to learn about patterns and their relation.*Most participants thought that VEISIG was useful to understand the patterns language and its relationships, so that it can help to understand better the language and its patterns. They learn with the recommendation how to extend their solution using expert's knowledge.
- *Users do not blindly trust the recommendations.* Even if the system might help in designing a solution, most of the users said they would have preferred to know the reasons that support such recommendations. Users prefer to know how a solution was designed instead of applying a solution whose source is unknown. Some authors like Donovan [5]argue that users are more comfortable with elements that are familiar so that black box systemsdo not favour trust. The VEISIG module could be considered as a black box system since it does not provide an explanation on the way recommendations are derived. However it also shows visually the relations amongst the recommendations and the selections done by the user, and, therefore, there is some implicit information on the closeness of recommendations and user selections which might explain why the participants found it useful.

5 Conclusions and Future Work

The use of design patterns may reduce development effort if designers are able to reuse the design knowledge. In this paper we haveintroduced a recommendation module that according to a preliminary study mighthelpto understand a design patterns language to novice users and it might produces produce better solutions or more complete solutions. Also we were able to reduce the design time, suggesting new solutions to the users.

The qualitative evaluation presented in this paper has shown several future works to undertake. Firstly, the recommendation process has to be more transparent to address a wider audience as said before. Users who want to learn to design need to trust on the recommendations by explicitly seeing the rationale behind them. A black box model of recommendation is not useful and users demands to improve their on the solutions more knowledge about how are obtained. Moreover, since design is a collaborative task this approach has to be extended to support cooperation amongst several designers. The validity of the experiment is limited and it should be repeated with real end-users to get sound conclusions and to detect how the recommendations could evolve to make up a really useful knowledge reuse environment for end-users.

Acknowledgments

This work is partly supported by the urTHEYgrant (TIN2009-09687) funded by the Spanish Ministry of Science and Innovation (MICINN).

References

1. Adomavicius, G., Tuzhilin, A.: Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering 17, 734–749 (2005)
2. Carroll, J.M., Farooq, U.: Patterns as a paradigm for theory in community-based learning. I. J. Computer-Supported Collaborative Learning 2, 41–59 (2007)
3. Díaz, P., Ignacio, A., Beth, R.M., Carroll, J.M.: A visual tool for using design patterns as pattern languages. In: Proceedings of the International Conference on Advanced Visual Interfaces, AVI 2010, Giuseppe Santucci, pp. 67–74. ACM, New York (2010)
4. Díaz, P., Rosson, M.B., Aedo, I., Carroll, J.M.: Web design patterns: Investigating user goals and browsing strategies. In: Pipek, V., Rosson, M.B., de Ruyter, B., Wulf, V. (eds.) IS-EUD 2009. LNCS, vol. 5435, pp. 186–204. Springer, Heidelberg (2009)
5. O'Donovan, J., Smyth, B.: Trust in recommender systems. Trust in recommender systems. In: Proceedings of the 10th International Conference on Intelligent User Interfaces, IUI 2005, pp. 167–174. ACM, New York (2005)
6. Weiss, M.: In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 711–723. Springer, Heidelberg (2003)

Managing Software Portfolios: A Comparative Study

Sebastian Draxler, Adrian Jung, and Gunnar Stevens

University of Siegen, Information Systems and New Media,
Hölderlinstrasse 3, 57068, Siegen, Germany
{sebastian.draxler,adrian.jung,gunnar.stevens}@uni-siegen.de

Abstract. Software applications that can be changed, modified and extended are nowadays pretty mainstream. But only few researchers focused on the role of the users social network for actual modifying practices and hurdles. Therefore this paper, studies in a comparative manner, how users modify software applications by using markets of existing components. We examine two popular applications: the universal tool platform Eclipse as an example for work applications and the game of World of Warcraft as an example for leisure applications. Despite the difference of the contexts, we found common patterns in collaborative actions within the social networks, that lead us to discuss the role of sharing and support for modification awareness for end users.

Keywords: EUD, Collaborative Tailoring, Software Ecosystems, Awareness.

1 Introduction

For a long time, research on managing software portfolios primarily focused on the appropriation of single applications. At a time, when applications had a clear border and when the software market was very limited this was well-suited. Examples are the work of Mackay [1] or Gantt and Nardi [2] who empirically investigated into tailoring efforts. One remarkable result of both studies was, how much collaboration in form of artifact or knowledge sharing they had observed. But since then the basic conditions have changed. Today it is often tried to establish so called *software ecosystems*. They consist of an open, extensible software platform that attracts different manufacturers and hobbyists, creating small-scale components, which can be individually assembled by end users [3]. Software ecosystems are an interesting topic to look upon, when it comes to study end user development. They empower the end user to choose or add functionality to their software by orchestrating pre-existing modifications. Compared to the situation of the 1990s, software ecosystems and the involved stakeholders are globally networked. We believe that this changes the users opportunities for modifying applications, since under these circumstances local networks of users (e.g. a company) collaboratively makes use of software ecosystems. Our first goal is therefore to understand: *"How and based on what information do people modify their personal software installations?"* To answer this question, we followed a similar approach as Mackay [1], which can be described as a set of empirical field studies, consisting of observations and interviews. Since we are not expecting an universal answer, we chose two quite different software ecosystems to investigate into, hoping for

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 337–342, 2011.

contrasting results. First we analyzed how professional software developers modify their Eclipse installations during their day to day work. This was followed by a second series of investigations, that examined how players of the online role playing game World of Warcraft (WoW) modify their game clients during leisure time.

2 Two Field Studies on Managing Software Portfolios

Our research process is loosely oriented on Mackay's [1] studies. For each study (Eclipse and WoW) we started exploring the relevant literature on the software, organizations and communities that are related to the case. This was followed by partially-structured interviews and in the case of Eclipse, on-site observations. The work is still carried out as open ended qualitative study. The material presented here was transcribed and the transcripts analyzed using coding mechanisms of the Grounded Theory [4] approach. While this is not a full grounded theory (so far we rely on In-Vivo codes), the approach has still proven very helpful to carefully analyze the material, without subsuming our observations under pre-defined categories from literature.

For the Eclipse case we cooperated with six software companies with 10 to 250 employees. At each company, we conducted at least two semi-structured interviews of at least one hour (altogether, we conducted 17 interviews. Additionally, we visited two of the smaller companies over a period of 3-5 days for on-site observation. For the WoW study, we interviewed a small WoW guild that is constituted by 8 to 10 active members of various game experience and with a different educational background. Whereby a guild is an in-game association of player characters formed to make the accomplishment of group-related tasks easier. We conducted at least two semi-structured interviews of 15 to 60 minutes with each player. Additionally, we recorded the changes in their addon configuration over a period of one month.

2.1 Customizing the Eclipse IDE (Study 1)

Eclipse is a multi-language integrated development environment (IDE) and an extensible software ecosystem. It began as a toolbox for the Java programming language at IBM. It was designed to integrate future tools under one roof, using a plug-in mechanism. The platform was made freely available, open source and steered by a non-profit foundation to attract other companies and other contributors.

Eclipse provides all of its functionality on top of a core runtime system and can be extended by using additional third-party plug-ins. The Eclipse core and most additional plug-ins come free of charge and are released under the terms of an open source license. An Eclipse plug-in is constituted by an XML description and Java code that supplies the functionality. There are about 900 tools available that consist of thousands of *plug-ins* called components. Tools are either installed using the included install and update mechanism or just copied to a folder manually.

Software developers as we interviewed and observed have to fulfill quite different tasks, from documenting requirements, modeling, coding, testing, debugging to talking to customers. For many of these tasks special tools are either needed or at least are a great help. Furthermore the resulting artifacts are often shared among other developers who work on the same or similar tasks. Eclipse allows to be extended by additional plug-ins that could support the task in question. Most of the observed and interviewed users were capable of creating such plug-ins on their own. But due to the amount of plug-ins that already exist on the global Eclipse ecosystem and the time and effort necessary to create a new plug-in, they chose to first search for existing alternatives that might fit their requirements.

One of the key findings of the study, when the need for a new tool arose, suitable recommendations regarding tool selection, installation, and configuration were sought out from co-workers who also found themselves in similar working contexts. We especially could observe this in environments where software developers organized themselves in agile teams. People did trust in their co-workers advice much more than in recommendations found on the Internet or in magazines. Within the observed companies, several related strategies were established or put into practice by accident.

If it was obvious that someone should be told about what plug-ins to install, e.g. if a new person joined a project team, either the plug-in names or even the whole set of artifacts were passed to that person. In case of a problem, people went to colleagues and just asked for advice which plug-in to pick or how to proceed if problems occurred. Unfortunately quite often it was not clear who could be an experienced colleague for a certain topic. As some Eclipse users were constantly trying to stay informed on plug-in related topics, they sometimes stumbled upon interesting news that could also be relevant for their colleagues. This was an interesting source for innovation for their colleagues. But as they did not consider chatting about new tools as a central part of their job, it was often unclear if news were important enough to be shared. On several occasions Eclipse users sat together at one machine to discuss a problem or how to proceed with a project or task. While doing so, the colleague did discover new icons in the Eclipse toolbar and so the topic moved to new plug-ins. The result was an exchange on new interesting plug-ins.

Overall we observed that Eclipse users tended to ensure personal information exchange on which plug-ins are interesting, how to install them and which problems can occur. On the other hand it was often unclear if general plug-in related news, tips, experiences should be spread and through which channels. Therefore this was often triggered by accident.

2.2 Customizing World of Warcraft (Study 2)

With more than 12 million subscribers the massive multiplayer online role-playing game World of Warcraft (WoW) is currently the world's largest game of this kind. WoW was developed by Blizzard Entertainment and released in November 2004. Three expansions for the game have been released since then, in addition updates of the game client are regularly released.

WoW is an interesting example for EUD in games, because it is possible for players to create their own addons for the game client that is used to play. Addons are constituted by describing metadata, XML documents describing the user interface and the functionality, which is written in the scrip language LUA. The development of addons is officially encouraged by providing the user with access to certain game API functions. There are currently over 5900 user-created addons [5] for the player to choose from. Addons are installed by downloading a code package from the internet and then placing it in a specific addon folder in the WoW installation.

The game is designed in a way that people work together in groups to accomplish complex tasks in the game world. Players organize themselves in guilds in order to simplify group building and represent social networks. Although the game is designed in a way that players do not necessarily need addons, they play an important role because they can enhance the players or groups performance. Addon functionality can range from displaying additional information that is helpful to the player to automating certain tasks or reorganizing the built-in chat function. Not every player benefits from certain addons as the usefulness of an addon depends on the role that the player seeks to fulfill in the game.

Throughout the whole interview study, every person had modified his/her game client using addons, although this was not a selection criterion. Most of the collaborative innovation process happens via in game chat and voice chat. All interviewees use an external tool called TeamSpeak to coordinate group-related tasks and having general discussions. TeamSpeak works similar to a Skype or a telephone conference where the players connect to a persistent server in order to talk to each other. One of the key findings of our study was that most of the addons are installed, based on recommendations from other guild members. These recommendations are seen as more or equally important as recommendations on the addon-related websites. The experienced players try to stay up-to-date by informing themselves on various WoW addon sites about updates for their current addons or new addons that could enhance their playing experience. As part of the study, we discovered several non-formal modifications practices.

If players want to accomplish certain tasks or face problems concerning certain game elements, this was often discussed with guild members, as helping other players is quite common. As part of this, players often received recommendations on what addons could simplify the completion of this task. If a problem with an addon or the game client arose, the other guild members were always asked first. Unfortunately, often it was not clear which guild member could be an expert for a certain addon or problem. The continuous use of voice chat benefits the virtual collaboration in a way that it can create virtual over-the-shoulder learning situations. For example in one specific case a player mentioned certain statistics about his character and another player asked where he could find this statistic. The first player then realized that this statistic was generated by an addon and recommended it. All interviewees use a third party tool, called Curse Client (see [5]) to install new and update existing addons. It was developed to help players in installing new addons and managing their current addon configuration, by providing them with an easy-to-use interface representation of a rich addon database. Information about new and interesting addons is spread verbally to players which are suspected to have an interest in these. New players usually get recommendations on certain addons they should install. These recommendations are usually given to them before they engage a difficult task with other guild members.

2.3 Discussion

A common phenomenon that we expected was that people tend to employ preexisting modifications rather then developing their own, hence the fact that certain people in both studies had the skills to develop them. The interviewees in both studies argued, that using pre-existing modifications saves them a lot of time. What we did not expect were the similarities at the practice level as well as the reasons behind certain actions.

But by comparing the two studies, we found more similarities. Despite the differences of the contexts (work vs. play) and heterogeneous user groups (very skilled vs. varying) the sharing behavior was very similar. In both cases people relied on the recommendations of their friends or co-workers more or as much as they relied on recommendations made by external people or websites. We classified these as follows:

#1 asking, because of a problem

People actively asked their friends or co-workers about their software configuration and which modifications they use. This happened mostly if a problem or a new and unknown task appeared. In both studies we traced this back to the belief that the co-workers or friends better understand each others context and therefore are a more reliable source of information.

#2 asking, triggered by accident

In several cases, people either accidently observed or discussed the use of an modification unknown to others. This triggered a need for awareness of what the colleague or friend is using and resulted often in a discussion to exchange experience on modifications.

#3 actively spreading the news

If people were actively informing themselves or by accident picking up news on modifications, they tried to spread this information further to (potentially) interested people in their near social network.

#4 actively introducing new workers/players

Sometimes a new person joins the game of WoW or a colleague joins a certain development project or even the company. In this case people did introduce the “junior” not only to the work/game, but also recommended certain modifications.

If we mirror these categories back to the several observations and interviews, they lead back to a lack of awareness within social networks as colleagues or guilds. Even more, there is a constant and latent need for this kind modification or EUD awareness. But since there was no support, it took certain points of interaction like a breakdown, beneficial accident or a complex task to bring the topic of modifications to the center of attention and create this sort of awareness.

3 Related Work

Mackay [1] as well as Gantt and Nardi [2] empirically investigated into the collaborative effects of tailoring, as sharing knowledge and artifacts. Our work is similarly structured but takes current developments into account. More recent work on collaborative tailoring and the related topic of software appropriation was especially done by Pipek and Kahler [6]. They did describe different shared scenarios (concerning usage, artifacts or infrastructure) that also lead to a need for awareness. While their work discusses this topic marginally, we wanted to focus more deeply on the role of awareness in collaborative EUD processes.

While existing research efforts investigated into organizations or closed groups as target for their research, we focus on groups, that act as social networks and are embedded in software ecosystems [3]. Therefore we can address appropriation efforts in environments where large numbers plug-ins/addons from 3rd parties already exist. This results in a different view on collaborative tailoring efforts and awareness.

4 Conclusion

Our research shows that people may trust in particular recommendations of local peers, regarding modifications as plug-ins or addons. Both, Eclipse and WoW users facilitate third party tools, to orchestrate and maintain their sets of plug-ins or addons. These tools all share the same basic features. Curse for WoW, as well as the Eclipse Marketplace or Yoxos for Eclipse represent central repositories of components to modify the application. These tools keep track of what a user installs, they help keeping addons or plug-ins up to date and ensure an installation that is not broken after modifying. On the other hand, they miss to reflect the collaborative nature of modifying software that creates a demand for addon awareness and recommendations.

In small local or remote groups as the ones we examined, we found plenty of incidents were breakdowns, accidents or planned intervention functioned as a trigger that revealed a need for awareness. This resulted in discussions and recommendations. This was just “the tip of the iceberg”, as we observed a constant and latent need for modification awareness in groups. And as there already exist tools to support some of the users EUD efforts, future research should suggest and evaluate possibilities to support *modification awareness*, as it could improve the reach of EUD.

References

1. Mackay, W.: Patterns of sharing customizable software. In: Proc. of the 1990 ACM CSCW, Los Angeles, California, United States, pp. 209–221 (1990)
2. Gantt, M., Nardi, B.A.: Gardeners and gurus: patterns of cooperation among CAD users. In: Human Factors in Computing Systems, Monterey, California (1992)
3. Bosch, J.: From software product lines to software ecosystems. In: 13th International Software Product Line Conference. CMU, San Francisco (2009)
4. Glaser, B.G., Strauss, A.L.: The Discovery of Grounded Theory: Strategies for Qualitative Research. Aldine Pub. (1999)
5. Curse.com: World of Warcraft Addons - WoW Addons - Curse.com (2011/03/14), `http://wow.curse.com/downloads/wow-addons/default.aspx`
6. Pipek, V., Kahler, H.: Supporting Collaborative Tailoring. In: End User Development. Springer, Heidelberg (2006)

MikiWiki: A Meta Wiki Architecture and Prototype Based on the Hive-Mind Space Model

Li Zhu[1], Ivan Vaghi[2], and Barbara Rita Barricelli[1]

[1] Department of Computer Science and Communication, Università degli Studi di Milano,
Via Comelico 39/41, 20139 Milano, Italy
{zhu,barricelli}@dico.unimi.it

[2] MIKAMAI s.r.l.,
Via Ceradini 12, 20121 Milano, Italy
ivan@mikamai.com

Abstract. This paper presents MikiWiki, a meta-wiki developed to prototype key aspects of the Hive-Mind Space (HMS) model. The HMS model has been proposed to share the visions of End-User Development and meta-design in collaborative online environment development. It aims to support cultures of participation and to tackle the co-evolution of users and systems. The model provides localized habitable environments for diverse stakeholders and tools for them to tailor the system under design, allowing the co-evolution of systems and practices. MikiWiki is aimed at supporting the exploration of opportunities to enable software tailoring at use time. Such an open-ended collaborative design process is realized by providing basic building blocks as boundary object prototypes, allowing end users to remix, modify, and create their own boundary objects. Moreover, MikiWiki minimizes essential services at the server-side, while putting the main functionalities on the client-side, opening the whole system to its users for further tailoring.

Keywords: HMS model, Meta-design, End User Development, Boundary Objects, Co-evolution, Habitable Environment, Wiki, MikiWiki, Mikinugget.

1 Introduction

Web 2.0, social media and advanced information technology are changing the role of end users and the way they are sharing and managing knowledge, and encourage cultures of participation [1]. Complex design projects also need to actively engage all the stakeholders in the design process. However, communication among stakeholders often breaks down due to differences in cultures, backgrounds and modes of communication. Moreover, the co-evolution of design communities and software systems [2] requires open software development environments to support emerging needs. To tackle these issues, the Hive-Mind Space (HMS) model, a meta-design conceptual model has been proposed [3][4]. It aims to provide a common meeting space to bring diverse stakeholder Communities of Practice (CoP) [5] together as a Community of Interest (CoI) [6] and to support their collaborative design in an evolving manner. The HMS model brings software engineers, domain experts and end-users together to

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 343–348, 2011.

collaboratively work on design projects. Each stakeholder specific design community in the HMS model is provided with a ‘habitable environment’ [7] that provides essential tools for the CoP to perform its use and design activities. This environment is localized to the CoP’s culture, role and digital devices in use [8]. To enhance communication among CoPs, the HMS model introduces a central communication channel serving as a boundary zone [9], where different CoPs can create, exchange, and cooperate around boundary objects [10]. Boundary objects are artifacts that mediate communication, for instance, sketches can be used as boundary objects among architects, clients and civil engineers for reasoning about design.

The HMS model derives from the Software Shaping Workshop (SSW) methodology [11] and supports three different levels of participation and design activities: i) meta-design level, where software engineers maintain the system and design environments for domain experts; ii) design level, where domain experts design environments for end users; iii) use level, where end users tailor and use the environments and tools. In addition, the HMS model has an open infrastructure, i.e. CoPs can tailor their habitable environments.

The HMS model has been applied before to two different cases, mechanical engineering and collaborative knowledge management for tourism [3][4]. In these two applications, the environments were designed and implemented using a specific software framework – the BANCO framework [8][12].

This paper describes MikiWiki, a new and still in-progress implementation of HMS, based on a new architecture and design principles.

2 Backgrounds and Related Work

End-User Development (EUD) is a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artifact [13]. A prominent motivation for EUD is addressing “design as a process, which is tightly coupled to use and continues during the use of the system” [14].

Nevertheless, EUD tends to solve the co-evolution [2] problem from a techno-centric perspective and mainly focuses on tailorability without taking other technology-related communication, demonstration and negotiation activities into consideration [15].

In the past few years, mashup software applications have been studied as possible tools to support end-user development activities [16]. Two popular examples of mashup applications are Yahoo Pipes [17] and Microsoft Popfly [18]. Yahoo Pipes is a tool for aggregation and manipulation of content available in the Web, while Microsoft Popfly (discontinued since 2009) was a tool for Web pages and games creation by visual composition.

MikiWiki merges the high potentiality of visual composition and direct manipulation interaction style with mashup techniques to empower end users and enable then to become end-user developer. Moreover, MikiWiki is based on the wiki [19] approach that fosters collaboration among users and supports their communication.

3 MikiWiki

MikiWiki is a prototype that implements the HMS model and allows to further explore EUD and meta-design approach [20]. However, the mapping of HMS concepts to MikiWikis mechanism is not one-to-one, as many theoretical concepts, such as boundary objects, cannot be reduced to a simple software system component. We started by implementing a simple wiki system, then extended it to support different levels of participation, boundary objects, habitable environments and, finally, the mediation mechanism. We chose a wiki approach to prototype the HMS model since wikis have an open editable structure, pages as basic units of sharing, existing documented architecture models and implementation of traditional wikis. All this makes them a good starting point for prototyping the HMS model.

3.1 Habitable Environments in MikiWiki

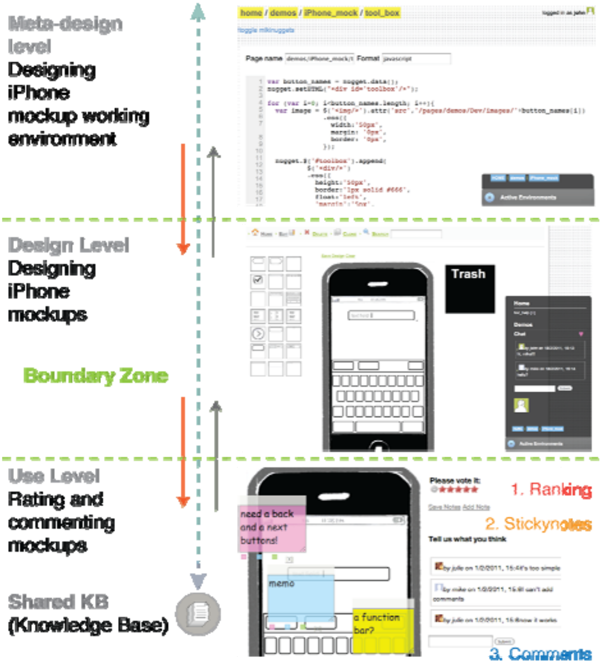
In accordance with the HMS model, a flexible mechanism is designed to allow CoPs to partition and locally configure communication [3]. Environments are used as a way to associate specific behavior to a large set of pages, for instance, by customizing an environment to a domain with specific tools and working habits [23] (Fig. 1).

Environments allow customization, without imposing predefined structure on all CoPs, but allowing the sharing among selected members. For example, access control in MikiWiki is not an inherent property of all environments, but it can be achieved by including a "check point" mikinugget in the environment settings page: all the pages within this environment inherit the access control property. Environments can be structured hierarchically: an environment inherits all the settings of the environments containing it. Individual environments can be further extended or override the properties of their parent environments.

Environments are also a mechanism to negotiate the awareness of the convergent and divergent viewpoints, by which diverse points of view from other environments are represented. Being aware of those differences eventually enhances the mutual understanding and supports CoPs collaboration. The communication and rendering mechanisms are also expressed via mikinuggets, thus the behavior and rendering of pages can be customized and extended. For example, an architectural plan visualized within an environment used by a Japanese construction team using iPads will require different representations of the same plan as seen by the German procurement department of the building company for whom the same data might be more meaningfully represented as a spreadsheet with building materials price list.

3.2 MikiWiki in Use

MikiWiki has been primarily developed to support diverse CoPs (mainly software engineers, designers and clients) collaboratively designing iPhone applications (Fig. 1). At the meta-design level, software engineers design workflow and interactions, which generate domain-oriented habitable environments for different CoPs. At the design level, designers use one of the environments to create iPhone mockup applications by simply using drag-and-drop components and sharing their designer results with all team members. At the use level, users can vote, annotate and discuss the mockups with the designers. This last environment consists of an iPhone mockup and three boundary objects used for ranking and adding sticky notes and comments. Mikinuggets are designed to support communication and negotiation among CoPs. For instance, design teams can easily create a shared to-do list to coordinate their collaboration, and add an online presence mikinugget to increase awareness.



```
var button_names = nugget.data();
nugget.setHTML("<div id='toolbox'/>");

for (var i=0; i<button_names.length; i++){
  var image = $('<img/>').attr('src','/pages/demos/Dev/images/'+button_names[i])
                .css({
                  width:'50px',
                  margin: '0px',
                  border: '0px',
                });

  nugget.$('#toolbox').append(
            $('<div/>')
            .css({
              height:'50px',
              border:'1px solid #666',
              float:'left',
              'margin':'5px',
```

by julie on 1/2/2011, 15:4it's too simple

by mike on 1/2/2011, 15:8I can't add comments

by julie on 1/2/2011, 15:8now it works

Fig. 1. iPhone mockup environment in MikiWiki

3.3 MikiWiki Architecture

In MikiWiki, the server side supports the minimal amount of features and services that maintain basic functionality. When possible choices should not be expressed as server-side code, but as a wiki page that gets executed on the client-side and that relies on server-side 'primitive' services invoked via AJAX techniques. Some functionality cannot be removed from the server side specifically all the basic facilities that handle the sharing of information, since the server acts as a repository and single aggregation point for wiki pages. As a concept demonstrator, MikiWiki aims to explore some key characteristics of meta-design, i.e. design infrastructure tailorability and EUD.

3.4 Mikinuggets and Open Infrastructure

Mikinuggets in MikiWiki are explicitly designed to reflect the HMS model's boundary objects' concept. A mikinugget is a page embedded within another page in order to create sharable remixable components. We provide a set of mikinuggets for instance ranking, commenting, annotations, drawing tools, notification, online presence, change-tracking, user-tracking, chat, to-do list, video embedding, access control and profile to be embedded.

The separation between user interface and application, the surface and the deep [21] restricts end users to simple manipulation of surface features, while the deeper system remains only accessible to developers. However, developers often do not know all the ways in which the system might be used by different end users over time [22]. Hence some lower-level details of system behavior should be also available for customization at the user interface.

Mikinuggets allow end users further appropriation of the system. Mikinuggets' pages act as a mechanism and interface for supporting the creation and evolution of software artifacts beyond their initial form. Moreover, mikinuggets are also a medium made of captured knowledge. CoPs can incrementally construct knowledge via mikinuggets during collaboration and communication. Non-programmers can start using and remixing existing mikinuggets, while advanced users can clone and modify these mikinuggets and consequently introduce new behaviors.

4 Conclusions

MikiWiki is a meta-wiki architecture and a work-in-progress prototype to support and evaluate the HMS model. It brings diverse CoPs together to participate in the design process, support their communication and evolve all the system components as well as the communities themselves. Considering situated innovation emerging in local contexts, MikiWiki aims to provide a just-enough infrastructure based on under-design principles, allowing users to further build, extend and develop their own environment. The next step will be to apply MikiWiki to various use scenarios, to improve its usability as well as to focus on how mikinuggets and habitable environments can be used to support and shape collaboration and communication.

Acknowledgments. The work of Li Zhu and **Barbara Rita Barricelli** is supported by the Initial Training Network "Marie Curie Actions", funded by the FP 7 - People Programme with reference PITN-GA-2008-215446 entitled "DESIRE: Creative Design for Innovation in Science and Technology."

References

1. Kolbitsch, J., Maurer, H.: The Transformation of the Web: How Emerging Communities Shape the Information we Consume. J. Universal Computer Science 12(2), 187–213 (2006)
2. Bourguin, G., Derycke, A., Tarby, J.C.: Beyond the Interface: Co-evolution inside Interactive Systems - A Proposal Founded on Activity Theory. In: Blandford, A., Vanderdonckt, J., Gray, P. (eds.) Proc. of IHM-HCI 2001, Toulouse, pp. 297–310 (2001)
3. Zhu, L., Mussio, P., Barricelli, B.R.: Hive-mind space model for creative, collaborative design. In: Proc. of DESIRE 2010, Lancaster, UK, pp. 121–130 (2010)

4. Zhu, L., Barricelli, B.R., Iacob, C.: A Meta-design Model for Creative Distributed Collaborative Design. IJDST 2(4) (2011)
5. Wenger, E.: Communities of Practice. Learning, Meaning, and Identity. Cambridge University Press, Cambridge (1998)
6. Fischer, G.: Communities of Interest: Learning through the Interaction of Multiple Knowledge Systems. In: Bjornestad, S., Moe, R., Morch, A., Opdahl, A. (eds.) Proc. of IRIS 2001, pp. 1–14 (2001)
7. Reenskaug, T.M.H.: The Model-View-Controller (MVC) - Its Past and Present. JavaZONE, Oslo (2003)
8. Barricelli, B.R., Marcante, A., Mussio, P., Parasiliti Provenza, L., Valtolina, S., Fresta, G.: BANCO: a Web architecture supporting unwitting end-user development. IxD&A - Interaction Design & Architecture(s): Design for the Future Experience 5-6, 23–30 (2009)
9. Andersen, R., Mørch, A.I.: Mutual Development: A Case Study in Customer- Initiated Software Product Development. In: Pipek, V., Rosson, M.B., de Ruyter, B., Wulf, V. (eds.) Proc. of IS-EUD 2009, pp. 31–49. Springer, Berlin (2009)
10. Star, S.L., Griesemer, J.R.: Translations and Boundary Objects: Amateurs and Professionals in Berkley"s Museum of Vertebrate Zoology, 1907-1939. Social Studies of Science 19(3), 387–420 (1989)
11. Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A.: Visual Interactive Systems for End-User Development: A Model-based Design Methodology. IEEE TSMCA 37(6), 1029–1046 (2007)
12. Barricelli, B.R., Iacob, C., Zhu, L.: BANCO Web Architecture to Support Global Collaborative Interaction Design. In: Proc. of IWIPS 2010, pp. 159–162. P&SI (2010)
13. Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-User Development: An Emerging Paradigm. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development, pp. 1–8. Springer, Dordrecht (2006)
14. Henderson, A., Kyng, M.: There's no place like home: Continuing Design in Use. In: Design at work: Cooperative Design of Computer Systems, pp. 219–240. Lawrence Erlbaum Ass., Mahwah (1991)
15. Pipek, V.: From tailoring to appropriation support: Negotiating groupware usage. PhD Thesis, University of Oulu, Oulu (2005)
16. Wong, J., Hong, J.: Making mashups with marmite: towards end-user programming for the web. In: Proc. of CHI 2007, pp. 1435–1444. ACM, New York (2007)
17. Yahoo Pipes, `http://pipes.yahoo.com/pipes/`
18. MicrosoftPopfly, `http://www.microsoft.com/nz/digitallife/internet/what_can_I_do_with_Popfly.mspx`
19. Fischer, G.: Social Creativity, Symmetry of Ignorance and Meta-design. Knowledge-Based Systems Journal 13(7-8), 527–537 (2000)
20. Leuf, B., Cunningham, W.: The Wiki Way: Collaboration and Sharing on the Internet. Addison-Wesley, Reading (2001)
21. Dourish, P.: Developing a Reflective Model of Collaborative Systems. ACM Transactions on Computer-Human Interaction 2(1), 40–63 (1995)
22. Greenberg, S., Marwood, D.: Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface. In: Proc. of CSCW 1994, pp. 207–217 (1994)
23. Mussio, P., Finadri, M., Gentini, P., Colombo, F.: A Bootstrap Approach to Visual Interface Design and Development. The Visual Computer 8(2), 75–93 (1992)

Part IV
Doctoral Consortium

Collaborative Modeling from an End-User Perspective

Alexander Nolte

Information and Technology Management, Institute for Applied Work Science,
University of Bochum, Universitätsstr. 150, 44780 Bochum, Germany
alexander.nolte@iaw.rub.de

Abstract. Current approaches to collaborative modeling mainly focus on facilitated modeling workshops. This approach proves to be very beneficial when big parts of processes have to be newly designed that require the interaction of multiple process stakeholders. However as modern businesses require flexible adaption of processes to changing environments inside and outside an organization, this approach is not always feasible. So my goal is to seamlessly integrate tight and loosely coupled interaction in collaborative modeling.

Keywords: end-user modeling, collaborative modeling, collaboration.

1 Introduction

My particular research interest can roughly be described as collaborative end-user modeling, with end-users including all people working in a socio-technical environment [7]. In this context modeling does not only apply to the creation and modification of graphical representations of software systems, but to representations of the whole socio-technical environment including business and work processes as well.

However transferring knowledge about processes into expressions according to a modeling notation is a difficult task even for skilled modelers. This task becomes even more difficult for end-users who are not used to apply this transfer on a regular basis. Additionally modeling software is mainly designed and built for modeling professionals, making it difficult to use for end-users during their everyday work. So my research will put strong emphasis in the creation of a modeling-environment in which end-users are able to directly contribute to collaborative process modeling.

But creating an easy to use interface is only part of what is necessary to directly integrate end-users into modeling. Socio-technical environments require the collaboration of various different process participants and stakeholders. Considering the creation of models for these environments strong emphasis has to be put on the ability to flexibly apply different types of collaboration, ranging from facilitated workshops to asynchronous dislocated scenarios (c.f. [6] for a more sophisticated distinction).

So creating a modeling-environment that enables end-users to directly participate in process modeling without solely relying on the help of facilitators and process experts can be described as my primary goal. This modeling-environment includes an easy to use interface as well as the ability to flexibly apply different modes of collaboration by incorporating means of communication and coordination.

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 351–354, 2011.

2 Background

Modeling business and work processes is a complex task. It requires the allocation of several roles to different tasks as well as to the technical equipment that is used. Because of the complexity it is reasonable to draft these models collaboratively by involving various stakeholders which serve as domain experts. This approach known as collaborative modeling [10] has been widely discussed in literature. Usually it incorporates a series of facilitated workshops in which all relevant process participants and stakeholders are brought together. During these workshops a facilitator manages the communication while an additional chauffeur operates a modeling tool[1].

This inevitably sequential approach forces idle times for the majority of the participants as only one person may speak at the same time. Additional idleness is enforced as not every person is knowledgeable about every part of the process. These forced idle times are likely to cause frustration among the participants [9]. So it appears reasonable to enable parallel development of model parts which also saves time [3].

A first approach into flexibly applying different modes of collaboration in collocated modeling workshops has been taken by Andersen and Richardson who propose a variety of different activities which they call scripts [1]. This approach however is still limited to synchronous collocated settings and requires facilitation throughout the whole process. Newer approaches – thus also being limited to collocated settings – challenge the need of a facilitator in collaborative modeling as they enable the participants to enhance and modify the process model on their own [12].

These findings provide a great opportunity for the improvement and sustainability of process models in general as creating a process and executing it during the daily work practice is not sufficient. Problems are likely to arise that “cannot be completely anticipated at design time” [5]. The very same thing applies to software design in the context of end-user development [5]. Modeling without the need of a facilitator also enables modeling on demand which is dearly needed as work processes have to be flexibly adjusted to changing conditions. This however cannot be done by entering the previously described process design workshops again, because these design cycles would be “too slow, time-consuming and expensive” as Lieberman states in the context of end-user development [8]. In order to increase the flexibility of modeling it may not be restricted to these kinds of workshops. It has to be integrated into the everyday work of the process participants. This is where collaborative process modeling and end-user development come together. Process participants have to be able to become co-designers [4] and create their own work-environment without the need for specially trained professionals [13].

3 Chosen Approach

There is very little evidence about which specific functions end-users require when using modeling tools in different collaborative settings[2]. So I chose an ethnographic

[1] c.f. [11] for a more exhaustive and sophisticated distinction of roles in collaborative modeling.

[2] c.f. section 1 for an elaboration on collaborative settings.

approach (c.f. [2]) because I wanted people to work in an environment that is familiar to them thus minimizing distraction.

Beforehand I conducted a first study which aimed at finding out whether end-users have the required skills to actively contribute to modeling. These include the ability to think in sequences and to translate thoughts into a modeling notation. As a start I involved future stakeholders – all but one not modeling experts – in a workshop where a process had to be designed from scratch. This process incorporates the work of a service agency offering accompaniment for elderly people during their weekly shopping. It is part of a three-year long interdisciplinary research project[3] which aims at enabling elderly people to live in their own home for as long as possible. During the course of this workshop the participants were ordered to contribute activities to a previously prepared process model. Each participant was given a laptop and contributed through a single text-input box in a web interface. Afterwards the participants were told to sort the contributed elements according to the process sequence. The actual moving of the process elements was conducted by a facilitator.

Results derived from this study show the ability of end-users contribute to process development, given adequate means to do so. However as moving elements inside the modeling tool could have been done by the participants themselves, there are possibly multiple other functions that process stakeholders can make use of. In order to find more of them before developing a prototype, I started with the ethnographic approach.

In order to incorporate the use of process models into the daily activities of non-process experts, multiple work process models were developed in different departments of two major steel fabricating companies in Germany. The participants that were involved in these workshops were predominantly metalworkers with no previous experience in modeling or the use of modeling software at all. We started by collaboratively modeling their work process in a fairly detailed manner. Afterwards the models were printed out to avoid any bias considering the use of graphical modeling tools. These printed models were attached next to the workbench of the metalworkers.

We advised them to pay close attention during their work to any irregularities compared to the standard process, visualized in the process model (like e.g. missing tools and material). They were advised to document these irregularities and attach the documents to the part of the model that represents the process step where the irregularity occurred. Once every week all employees of this specific department came together, trying to find the source of the problems that were documented during the week. Afterwards the paper based process model was collaboratively altered to fix the source of the problem. During these meetings I was able to monitor their activities.

First results confirm the results from the previous study: End-users are able to think in processes and have no problem identifying process parts where irregularities occur. Additionally they alter models by deleting elements (crossing them out) and providing additional information through annotations. These findings lead to a more sophisticated set of functions for an end-user friendly collaborative modeling tool and will be used to develop a first software prototype. This software will then replace the previously described paper based prototype in order to evaluate these functions. The prototype will also be tested in different other collaborative settings such as collocated modeling workshops, in order to ultimately come up with a suitable software

[3] Funded by the German Federal Ministry of Education and Research, FKZ 01FC08008.

tool and a surrounding environment that enables the seamless integration of tight and loosely coupled interaction in collaborative modeling.

References

1. Andersen, D., Richardson, G.: Scripts for group model building. System Dynamics Review 13(2), 107–129 (1997)
2. Bortz, J., Döring, N.: Forschungsmethoden und Evaluation für Sozialwissenschaftler. Springer, Berlin (1995)
3. Dennis, A., Hayes, G., Daniels Jr., R.: Re-engineering business process modeling. In: Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences, pp. 244–253 (1994)
4. Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A., Mehandjiev, N.: Meta-design: a manifesto for end-user development. Communications of the ACM 47(9), 33–37 (2004)
5. Fischer, G.: End-user development and meta-design: Foundations for cultures of participation. In: End-User Development, pp. 3–14 (2009)
6. Grudin, J.: Computer-supported cooperative work: history and focus. IEEE Computer 27(5), 19–26 (1994)
7. Herrmann, T.: Systems Design with the Socio-Technical Walkthrough. In: Handbook of Research on Socio-Technical Design and Social Networking Systems, pp. 336–351. Idea Group Publishing, Hershey (2009)
8. Lieberman, H., Paterno, F., Klann, M., Wulf, V.: End-user development: An emerging paradigm. In: End User Development, pp. 1–8 (2006)
9. Prilla, M., Nolte, A.: Fostering self-direction in participatory process design. In: Proceedings of the Eleventh Conference on Participatory Design 2010, pp. 227–230. ACM, New York (2010)
10. Renger, M., Kolfschoten, G., de Vreede, G.-J.: Challenges in collaborative modelling: a literature review and research agenda. International Journal of Simulation and Process Modelling 4, 248–263 (2008)
11. Richardson, G., Andersen, D.: Teamwork in group model building. System Dynamics Review 11(2), 113–137 (1995)
12. Rittgen, P.: Collaborative Modeling: Roles, Activities and Team Organization. International Journal of Information System Modeling and Design (IJISMD) 1(3), 1–19 (2010)
13. Sutcliffe, A., Mehandjiev, N., et al.: End-user development. Communications of the ACM 47(9), 31–32 (2004)

Designing in Use: Bridging the Design Time – Use Time Divide

Monica Maceli

Drexel University, College of Information Science and Technology
3141 Chestnut St, Philadelphia, Pa 19104 USA
Monica.Maceli@drexel.edu

Abstract. Meta-design theory emphasizes that future use can never be entirely anticipated at design time, as users shape their environments in response to emerging needs. Systems should therefore be designed to adapt to future conditions in the hands of end users, empowering end-user development to take place in a continuous, participatory manner. In our increasingly complex technological environments, tomorrow‘s meta-designers must be able to anticipate the environment in which the end users will work in order to provide the flexibility for users to craft and develop their tools. By exploring and projecting forward current trends in technology use, I have identified key principles for meta-designers and suggest that using them as design heuristics will aid meta-designers in crafting systems for future end-users to employ in designing and developing their future environments. This paper describes my doctoral research, aimed towards validating and critiquing these meta-design principles.

Keywords: design, meta-design, design time, use time, heuristics, context.

1 Introduction

At one time, design and use were closely entwined activities: *human crafters* designed tools through use and there was no distinctly separate design process. As technology advanced, industrialization introduced a divide between the goals of the setting of design (*design time*) and the setting of use (*use time*). Design time focused on experts creating a completed design artifact, while use time was oriented towards gradual user-driven evolution and change, responsive to environment and context. This tension between what could be accomplished at design time and what unpredictable situations the system would encounter during use has been an ongoing challenge to the evolving field of Human-Computer Interaction (HCI).

When environments of use were constrained to the workplace, our early HCI methodologies could strive to match known work tasks with suitable interfaces; this *human factors* approach focused on interfaces to afford interaction between man and machine. As technology moved into the home and into more complex environments of use and practice, HCI methodologies began to take a broader view of interaction, supporting *human actors* who controlled the technologies used in their daily lives [1].

However, recently developed technologies have allowed for complex and shifting contexts of use [2] as well as empowered users to design their own technological

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 355–358, 2011.

environments. Novel means of information and technology production (e.g. open source software development, mash-ups, commons-based peer production [3]) have radically changed the technological landscape. Users are again behaving as *human crafters* – controlling, designing, and developing not only their relationships with technology, but the very form and function of this technology.

2 Meta-design: Moving towards "Designing in Use"

The increasing evidence of "designing in use" behavior by end users is poorly supported by our existing HCI design methodologies which distance designers, both in time and space, from future scenarios of use and future users. Our design processes, in the words of Stewart Brand, "over-respond to the immediate needs of the immediate users" [4]. As Suzanne Bødker notes, there are currently many related challenges facing the field of HCI: (1) people need to be involved in design, not just as workers, but as someone who brings their entire life experience into the design, (2) this will necessitate a change in the way we design and prototype, and (3) we need to move away from end-user programming in isolation to configurations involving multiple people and multiple systems [2].

These challenges have been explored for many years in *end-user development* research, which seeks to empower end-users to design solutions to their problems in use. Fischer's recent work in this area [e.g. 5] suggests moving towards a future state of end-user development or *meta-design,* emphasizing *participatory co-design* throughout the life of the system. Meta-design describes a future state of design consisting of open systems that evolve during use, with design activities redistributed across time and levels of interaction with the environment. The framework emphasizes that the design of socio-technical systems must support flexible and evolving systems, that are not (and cannot be) completely designed before use, and that evolve in the hands of their users. Building on the great deal of previous research in end-user development, meta-design begins to place these ideas within a conceptual framework that can guide the successful creation of future socio-technical systems facilitating end-user development.

However, these ideas need further exploration to provide generalizable design methods to the HCI community, in an age of rapidly evolving and changing technology. My dissertation research, which is currently in progress, seeks to understand: *how should we design for a world that is increasingly full of human crafters?* Specifically, I seek to derive useful heuristics from key literatures and perspectives supporting systems that evolve in the hands of their users over their entire lifespan, exploring them first in a controlled laboratory setting and then on real-world design problems. These experiments are described in further detail within the following sections.

2.1 Principles for Designing in Use

The proposed idea generation process consists of a series of guidelines aimed at focusing thought towards common emergent behaviors that users engage in over time. These center around: *connecting* – to people with similar interests or needs, *reaching* *out* – in real-time across space and time, *combining* – the system with other tools and systems they use, *getting up to speed quickly* – so undue time is not spent learning the system, and *tailoring* – such that the system is molded to their personal needs. The rationale behind the inclusion of each guideline is briefly described below:

Guideline 1: ***Connect*** *with other people with similar needs and interests, both nearby and far away.*

John Thackara's [6] series of design frameworks for complex worlds emphasizes the increasing importance of systems that allow people to connect and communicate both locally and across the boundaries of time and space. This guideline intends to encourage these possibilities by focusing designers on how users can use the system to connect to similar people or extend the system in this direction.

Guideline 2: ***Reach out*** *and converse with other people in real-time, while they are using the system.*

Research prior to meta-design has explored modifiable systems that allow for reflective use-time conversations to occur, between designers and users [e.g. 7]. This guideline seeks to emphasize how users can have live experiences and conversation with other people within, or around, the system.

Guideline 3: ***Combine*** *it with other tools and systems they use regularly.*

The new (or redesigned) system may be only one of several tools and systems they use on a daily basis or even at the same time. While designers can never anticipate exactly how their system might be used, a focus on the surrounding edge and combinatory effects may spark new ideas [6].

Guideline 4: ***Begin using it quickly****, without a lot of help or instruction.*

Alexander's unselfconscious culture of design [8] requires systems users can understand relatively quickly and then contribute to confidently. This guideline is oriented towards envisioning ways in which novice users could begin using systems quickly and confidently, potentially becoming empowered to act as designers.

Guideline 5: ***Tailor it*** *to their personalized needs.*

Henderson and Kyng's [9] early writings on designing in use identified tailorability as essential to systems supporting users acting as designers. The system may tailor itself to the particular individual's needs automatically or through the user's tailoring actions.

3 Current and Future Work

The proposed guidelines are currently being used in a series of experiments testing the ability of the guidelines to focus and encourage design-time idea generation. The intention of this work is to move the conceptual framework of meta-design and support for end-user development to the forefront of design discussions, in a form that is accessible both to designers and end users. By identifying key behaviors that users are motivated to engage in, independent of technology, the guidelines intend to focus designers on possibilities for how they can endow their system with *tools for future end-user development* in the hands of their eventual users.

The guidelines will first be explored in a controlled laboratory setting to ensure that they are well-understood and relevant to both users and designers. Then the guidelines will be used in design exercises to brainstorm features for an existing system (the Internet Public Library's website), involving both the official system designers and several end users. These exercises will be conducted multiple times, to begin to explore these issues more longitudinally, and the end users will be engaging in diary-keeping, noting when and where they might want to make changes as they use the system. This will contribute an understanding of what modifications users might want to do in use and their motivations. Additionally, from the designer perspective, it will help us understand what designers can focus on at design-time to move towards becoming true *meta-designers* oriented towards what changes users might consider in use as opposed to simply providing a "completed" system.

The longer term goal of this work is to introduce meta-design concerns into the common methodologies of HCI, going further than addressing immediate needs (for example, through usability testing) and towards considering more possibilities for future system use. As emphasized by meta-design, we can never anticipate all the future scenarios of use. However, the field of HCI needs to evolve its methodologies to account for continuous, emergent use and change. It is my hope that the guidelines, integrated into a participatory and continuous design process, can help designers provide the necessary tools for the future *human crafters* of the system.

References

1. Bannon, L.J.: From human factors to human actors: the role of psychology and human-computer interaction studies in system design. In: Greenbaum, J., Kyng, M. (eds.) Design At Work: Cooperative Design of Computer Systems, pp. 25–44. L. Erlbaum Associates, Hillsdale (1991)
2. Bødker, S.: When second wave HCI meets third wave challenges. In: Proceedings of the 4th Nordic Conference on Human-Computer Interaction: Changing Roles. ACM, Oslo (2006)
3. Benkler, Y.: The Wealth of Networks. Yale University Press, New Haven (2006)
4. Brand, S.: How Buildings Learn. Viking, New York (1994)
5. Fischer, G., Giaccardi, E.: Meta-Design: a framework for the future of end user development. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development, pp. 427–458. Springer, Dordrecht (2006)
6. Thackara, J.: In the Bubble: Designing in a Complex World. MIT Press, Cambridge (2005)
7. Fischer, G., Lemke, A.C., Mastaglio, T., Morch, A.I.: Using critics to empower users. In: Conference Using Critics to Empower Users, pp. 337–347. ACM, New York (Year)
8. Alexander, C.: Notes on the synthesis of form. Harvard University Press, Cambridge (1964)
9. Henderson, A., Kyng, M.: There's No Place Like Home: Continuing Design in Use. In: Greenbaum, J., Kyng, M. (eds.) Design At Work: Cooperative Design of Computer Systems, pp. 219–240. L. Erlbaum Associates, Hillsdale (1991)

Design Patterns in the Design of Systems for Creative Collaborative Processes

Claudia Iacob

Department of Computer Science and Communication, University of Milan,
Via Comelico, 39/41, 20135, Milan, Italy
iacob@dico.unimi.it

Abstract. Creative collaborative processes are more than often characterized by the synchronous collaboration of stakeholders with different backgrounds and expertise. However, little work has been done in identifying design patterns for the design of software systems which support such collaboration. This line of research aims at identifying such design patterns following a two-phase pattern mining process: 1). the analysis of the results of a series of design workshops during which participants would design applications for synchronous collaboration, and 2). the analysis of a set of software applications which support synchronous collaboration in drawing, text editing, searching, and games.

Keywords: Design patterns, creative processes, synchronous collaboration.

1 Introduction

Creative collaborative processes bring together stakeholders with different backgrounds and expertise which communicate and interact with each other in real-time through activities such as searching [1,2], or sketching [3]. However, little work has been done in identifying design patterns for the design of software systems which support common synchronous collaborative activities such as drawing, searching, text editing, or games. This work aims to identify a collection of design patterns to be used in the design of synchronous collaborative systems. The patterns address communities of both novice and experienced software designers interested in applying a set of documented best practices in their design processes and in being supported in the collective understanding of problems and solution alternatives.

The concept of design patterns was first introduced in the ‘70s by Alexander [4], who proposed a pattern language for architectural design. Later on, the concept was adopted in domains such as software engineering and HCI [5]. In [6], a survey of 21 HCI pattern languages published between 1996 and 2007 shows that these collections target web user interface design, interactive exhibits, user interface related programming, hypermedia applications, or ubiquitous computing. Moreover, in the past few years, several collections of patterns have been proposed for the design of social interfaces [7], groupware technology [8], and cross-culture collaboration [9].

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 359–362, 2011.

2 Research Goals and Methods

The research question addressed by this work is: *What design patterns can be identified in the design of software systems which support creative collaboration in real-time?* To answer this question, a two-phase design pattern mining process was followed. During the first phase, a series of design workshops were conducted in order to identify the recurring design issues designers would consider in the design of software applications for synchronous collaboration. During the second phase, a set of 20 software applications which support synchronous collaboration were analyzed in order to identify those design issues considered in the implementation of concrete cases of applications. For both phases, 4 domains were considered for collaboration: drawing, text editing, database querying (searching), and games.

2.1 Design Workshops

A design workshop provides a team of 3-5 designers (the participants) with a set of problems, relevant to the area the mining process addresses. Each workshop lasts for approximately 2 hours and has 3 phases.

During the first phase, participants are encouraged to choose one problem from the set and to define as many scenarios [10] as they can consider for software solutions (applications) to tackle the problem. The second phase asks participants to choose another problem from the list and to find similarities and differences between the two problems (the one chosen during the first phase and the one chosen during the second phase). Lastly, participants are asked to design the GUI and the interaction process of the application related to the problem they initially chose during the first phase. For that, they are strongly encouraged to sketch their ideas, express all the design problems they encounter and, possibly, create a mock up of their overall design. A facilitator is present during each workshop and his/her role is to introduce the participants to the topic of the workshop and observe them, taking notes of their conversations.

13 teams - including professional graphic designers (8%), graduate (72%) and undergraduate (20%) students in Computer Science - participated in design workshops which addressed 5 problems subject to synchronous collaboration: drawing, text editing, database querying, puzzle solving, and crosswords solving. The design issues discussed by the teams were collected after each workshop. For each issues, its degree of recurrence (DoR) was computed as the ratio between the number of workshops during which the design issue has been discussed and the total number of workshops conducted.

2.2 Collaborative Software Analysis

As a second phase of the design pattern mining process, a set of 20 software applications which support synchronous collaboration has been analyzed through scenario and software walkthroughs. The goal of this analysis was the identification of those design issues considered in the implementation of concrete cases of applications and had as starting point the list of design issues identified during the workshops. For each of these design issues, its degree of recurrence was computed, similarly as in the case

Table 1. The most recurring design issues in the software analysis

Design issue	Synergo [3]	CoSearch [1]	GoogleDocs [11]	Coagmento [2]	M@M [13]	CPG [14]	TellTable [12]	DoR
Who is the coordinator in a real-time collaboration? How do collaborators coordinate themselves?	🟩	🟩	🟩	🟩		🟩	🟩	85.71
Support the visualization and the storage of the history of the collaboration	🟩	🟩	🟩	🟩			🟩	71.42
Support the tracking of each collaborator's individual contribution		🟩	🟩	🟩	🟩		🟩	71.42
Support collaborators' communication; integrate instant messaging features in the application	🟩		🟩	🟩	🟩			57.14
Visualize what others are doing in real time		🟩	🟩	🟩		🟩		57.14
Adapt application to several devices; make sure collaborators using different devices can work together in real time		🟩			🟩	🟩		42.85
Support collaborators in providing feedback, comments, rankings	🟩	🟩		🟩				42.85
Design the application for the web		🟩	🟩				🟩	42.85

of the design workshops, as the percentage of recurrence of the issue in the implementations analyzed. This paper reports on the partial results obtained through the analysis of a limited number of application (Table 1).

3 Partial Results and Expected Contributions

The design issues with the highest DoR in both the workshops' results and the software analysis results were considered for being documented through design patterns. This subsection briefly describes a subset of the patterns identified, their full description being provided in [15].

Who is the coordinator? addresses the problem of providing a coordination mechanism which: a). allows all collaborators to take part in the collaborations and b). maintains the resource in a consistent state at all times.

Integrated chat suggests integrating an instant messaging feature in the design of the application in order to support communication among collaborators.

Eyes wide open addresses the problem of allowing each collaborator to be notified about and visualize what the others are contributing to the process at any time.

Choose your collaborators suggests allowing each user to be able to choose the people s/he wants to work with during the collaboration.

Collaboration, always social suggests integrating mechanisms of tagging, ranking, annotating, and commenting in the application in order to support the collaborators in forming a community.

With or without collaboration addresses the issue of providing users with an additional private area, not available to the other collaborators, where each collaborator has total control and where s/he is provided with tools specific to the context of the application.

This work will provide a better understanding of the design issues to be faced during the design of synchronous collaborative systems by providing as output a pattern language to be used for the design of synchronous collaborative applications.

References

1. Amershi, S., Ringel Morris, M.: CoSearch: a system for co-located collaborative web search. In: Proceeding of CHI 2008, pp. 1647–1656. ACM, New York (2008)
2. Shah, C., Marchionini, G., Kelly, D.: Learning design principles for a collaborative information seeking system. In: Proceedings of CHI 2009, pp. 3419–3424. ACM, New York (2009)
3. Margaritis, M., Avouris, N., Kahrimanis, G.: On Supporting Users' Reflection During Small Groups Synchronous Collaboration. In: Dimitriadis, Y.A., Zigurs, I., Gómez-Sánchez, E. (eds.) CRIWG 2006. LNCS, vol. 4154, pp. 140–154. Springer, Heidelberg (2006)
4. Alexander, C.: A pattern language: Towns, buildings, construction. Oxford University Press, New York (1977)
5. Borchers, J.: A Pattern Approach to Interaction Design. John Wiley & Sons, Inc., Chichester (2001)
6. Kruschitz, C., Hitz, M.: Analyzing the HCI Design Pattern Variety. In: Proceedings of AsianPLoP 2010, GRACE-TR-2010-01 (2010)
7. Crumlish, C., Malone, E.: Designing Social Interfaces. O'Reilly Media, Inc., Sebastopol (2009)
8. Lukosch, S., Schümmer, T.: Communicating Design Knowledge with Groupware Technology Patterns: The Case of Shared Object Management. In: de Vreede, G.-J., Guerrero, L.A., Marín Raventós, G. (eds.) CRIWG 2004. LNCS, vol. 3198, pp. 223–237. Springer, Heidelberg (2004)
9. Schadewitz, N.: Design Patterns for Cross-cultural Collaboration. International Journal of Design 3(3) (2009)
10. Carroll, J.M.: Five Reasons for Scenario-Based Design. In: Proceedings of HICSS 1999, p. 3051. IEEE Computer Society, Washington, DC, USA (1999)
11. GoogleDocs, `http://www.docs.google.com`
12. Adler, A., Nash, J.C., Noel, S.: Evaluating and implementing a collaborative office document system. Interact. Comput. 18(4), 665–682 (2006)
13. Klopfer, E., Perry, J., Squire, K., Jan, M.F., Steinkuehler, C.: Mystery at the museum: a collaborative game for museum education. In: Proceedings of CSCL 2005. International Society of the Learning Sciences, pp. 316–320 (2005)
14. Battocchi, A., Pianesi, F., Tomasini, D., Zancanaro, M., Esposito, G., Venuti, P., Ben Sasson, A., Gal, E., Weiss, P.L.: Collaborative Puzzle Game: a tabletop interactive game for fostering collaboration in children with Autism Spectrum Disorders (ASD). In: Proceedings of ITS 2009, pp. 197–204. ACM, New York (2009)
15. Iacob, C.: Mining for Patterns in the Design of Systems for Synchronous Collaboration. In: Proceedings of AsianPLoP 2011 (2011)

EUD in Enterprise Open Source Learning Environments

Alessandro Pagano

Università degli Studi di Bari, Dipartimento di Informatica, via Orabona 4, 70125 Bari, Italy
alessandropagano@di.uniba.it

Abstract. Open Source development model gives users the opportunity to contribute. The development of Open Source System does not end at deployment time but requires continuous user participation and contribution. Many companies are involved in Open Source communities for enterprise software development and a huge amount of investments are related to training and new paradigms of Distance Learning. Open Source Community approach in Learning Environments code development, in Open Source courses and Learning Objects gives each user (and each company) the opportunity to contribute to software and course development. The new e-learning frontier and the world research community interest are focused on collaborative learning approach. Using Intelligent agent and systems (based on reasoning tools or linked data complex querying) makes possible to give each learner or teacher the opportunity to build customized learning paths. Learning Intelligent System (LIS) should recommend learning objects based on learner skills and learner final goals.

Keywords: Open Source, Learning Environments, SCORM, Learning Intelligent System, Linked Data, Ontology.

1 Research Background: Open Sources Driven Enterprises in e-Learning Fields

The Free and Open Source Software movement has had phenomenal impact on industry evolution: most of companies today, makes extensive use of Open Source Software (OSS) and technologies. Research, academic and professional communities are engaged in the study of the open software-related development in order to highlight advantages and disadvantages in terms of technology, re-use and economic impact.

An Open Source software could be adopted for ideological or purely pragmatic reasons [1], but it is necessary that the strategic management takes constant critical points that may influence this choice: a decision not weighted, it can cause damages to the company, or make them miss opportunities in terms of profit and development. The variables involved are numerous and not always immediately visible at a first analysis.

The Open Source development model gives users the opportunity to contribute in various projects. Open Source systems development does not end at deployment time but requires continuous user participation and contribution.

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 363–366, 2011.

One critical factor that enables the continual evolution of an OSS project is the forming of a vibrant and sustained community of practice [2] of developers, users and user-turned-developers. In OSS projects, there is no clear distinction between developers and users: all users are potential developers, everyone at his own level of knowledge and technical skills.

For an OSS project to have a sustainable development, the system and the community must co-evolve. A large base of voluntarily contributing members is one of the most important success factors of OSS. The role that an OSS member plays in the community is not pre-assigned, and is assumed by the member as he interacts with other members [3].

The approach described [4] is quite different when a company decides to invest in Open Source Software development, in fact facing the community, a company could play different roles:

- Open code indifferent: in this scenario, the source code availability is neither an advantage nor a disadvantage for the organization. OSS serves as a black box and its advantages or disadvantages are comparable to proprietary packaged software.
- Open code scholar: in this scenario, the organization considers the source code availability to be an advantage, but it does not use it to study or customize the program. Some organizations choose OSS because they feel that the program is less likely to contain hidden features or bugs, and that in case of bugs discovery, it will be fixed quickly. With this kind of approach the use of OSS implies a learning process, in which the organization gains experience and skills (and this could represent a profitable investment)
- Open code developer: in this scenario, OSS serves as a white box [5]. Organizations use the source code to study the software inner workings or to adapt the software to their own (or their clients) needs. This is primarily interesting for software houses developing OSS-based applications.

In OSS approach there is an upside not present often in proprietary software: anyone can verify the quality of the software code because the source code is available to everyone [6].

2 Student Generated Learning Path for Powerful Learning Experience

The research in e-learning field is focused on automatic (or semi-automatic) building learning path for each learner. The idea is that a learner could select and customize his own learning path guided by an intelligent system that suggests and recommends learning objects based on learner experience and previous skills. The final goal is to reach a new level of competence or skills through customized learning paths.

Students and teachers in this learning environments are the end user developers. They could design their own learning path aided by Learning Intelligent System.

The research approach to achieve this goal is composed by 6 main steps:

1. Provide a best practice analysis report about open source learning environments and identify reference international standards for e-learning (for example: Learning Objects, learning paths, Ontologies, Metadata, Linked Data);

2. Provide a common framework in order to design Learning Objects and learning Path;
3. Design and development of LIS (Learning Intelligent System) according to previously defined standards;
4. Develop a bridge between learning repositories and learning management system;
5. Implement courses on web portal and test in a pilot course;
6. Evaluation of results.

Actually the research team is involved in LIS design and development. The crucial point is to identify the best semantic approach for this specific situation.

In order to create dynamic learning paths it becomes crucial to model users profiles and correlate them to a suitable competency model. The idea behind this approach is to exploit actual vocabularies and ontologies that can be used to capture metadata about people and competencies using Semantic Web technologies. Technically this goal is achieved using linked data or ontologies.

3 Ontologies vs. Linked Data

This will lead to the application of common machine-readable formats (e.g. RDF) as well as enabling technologies (e.g. SPARQL) in order to implement practical use cases for the generation of customized learning paths.

Recently some principles that go under the name of Linked Data defines some good practices on how to expose, share and connect data on the Semantic Web. These important recommendations contribute to interoperability aspects of Semantic Web applications through advocating common guidelines.

The principles underlying this vision of Web of Data gave rise to the Linking Open Data Project [7] which is a community-led effort to create openly accessible, and interlinked RDF Data on the Web. When this community began its efforts, at the end of 2007, there were only a handful of these connected, and openly accessible, sets of data. Today, the Linked Open Data graph [8] shows that an increasing number of data providers have begun publishing data using these principles, leading to the creation of a Web of Data that already contains billions of statements.

Linked Data can provide valuable means to publish and connect structured learning objects on the Web when developing course materials and specifying global curricula in the form of learning pathways. An Intelligent System will use a repository of Learning Objects compatible with the SCORM standard, facilitating the interoperability of training materials with all the systems that support the standard. The Learning object builder (the “expert”) will lead steps on creating linked SCORM Learning Objects that will be automatically represented according to the Linked Data guidelines. In this context, LIS, through its plug-in in the LMS will discover new relationships using open linked datasets available on the Web of Data and will propose to the student an appropriate learning path in order to achieve training goals. LIS needs a powerful and scalable server structure to run efficiently: a web host side to host the entire platform, and a develop/services side to run the intelligent agent, based on J2EE platform.

Learning Objects, ontologies, RDF stores and all supporting data, which are essential for the right functioning of LIS, will reside in internal specific repositories and database, and represent the knowledge on which to base logical inference. Furthermore, the agent will communicate with the repository of Learning Object SCORM compliant to obtain additional materials. The LMS, which are installed on separate servers, can use the features of intelligent agent communicating through web services interfaces. LIS features will be presented in a transparent way to final user through a plug-in installed in the LMS used by the user. In fact, the final user (teacher and student) who wants to use LIS features will just need a web browser to join his open source LMS: it will be the plug-in to communicate through web services with LIS.

Finally, the creator of Learning Object will direct join LIS through HTML interface specially developed that will help him to enter Learning Objects and knowledge. The system will deliver Open Courses according to defined didactic structure. The didactic project is a full path that follows the learning goals and aims at the main theme divided into disciplinary segments represented by individual Modules and Teaching Units. The project is the foundation and the framework of the learning path offered to the student, delimiting the area of the formative intervention. The didactic project drafting is the production of all didactic items useful to the creation of the course following a defined standard. Such items are provided in the first drawing from experts and then worked by the staff in order to be used by LIS and other platform components.

References

1. Ven, K., Verelst, J., Mannaert, H.: Should you Adopt Open Source Software? IEEE Software, 54–59 (2008)
2. Wenger, E.: Communities of Practice – Learning, Meaning and Identity. England Cambridge University Press, Cambridge (1998)
3. Ye, Y., Fischer, G.: Designing for Participation in Sociotechnical Software Systems. In: Stephanidis, C. (ed.) HCI 2007. LNCS, vol. 4554, pp. 312–321. Springer, Heidelberg (2007)
4. Fischer, G., Piccinno, A., Ye, Y.: The Ecology of Participants in Co-evolving Sociotechnical Environments. In: Forbrig, P., Paternò, F. (eds.) HCSE/TAMODIA 2008. LNCS, vol. 5247, pp. 279–286. Springer, Heidelberg (2008)
5. Weinstock, C.B., Hissam, S.A.: Making Lightning Strike Twice. In: Feller, J., et al. (eds.) Perspectives of Free and Open Source Software, pp. 143–159. MIT Press, Cambridge (2005)
6. Pykalainen, T.: Model for profiting from software innovation in the new era in computing. Technovation 27, 179–193 (2007)
7. http://esw.w3.org/SweoIG/TaskForces/CommunityProjects/LinkingOpenData
8. http://richard.cyganiak.de/2007/10/lod/

Own Your Energy – Motivating People to Use Energy More Efficiently through Meta-design Environments and Cultures of Participation

Holger Dick

University of Colorado, UCB 430,
80309 Boulder, CO, USA
Holger.Dick@Colorado.Edu

Abstract. For my PhD research, I am investigating how meta-design software systems and cultures of participation can be used to motivate people to use energy more efficiently. My research is based on and extending two theories from social psychology and computer science, namely Psychological Ownership and Cultures of Participation. Combining these with the meta-design framework, I am building an integrated system in which users are being motivated to use energy more efficiently by supporting them to take an active role in the relevant decision processes.

Keywords: meta-design, cultures of participation, psychological ownership, social environments, energy.

1 Introduction / Research Problem

There is now overwhelming evidence that our current lifestyle is not sustainable and human energy consumption causes global warming [1]. To reduce energy consumption to sustainable levels, technological innovations and policy changes are likely not sufficient – a change in human behavior is necessary. While there has been recent work on combining Human-Computer Interaction and sustainability, these efforts are mostly focusing on immediate feedback and short-term changes, without showing any long-term changes in behaviors.

In my PhD research, I am developing and evaluating a software system to answer my guiding research question how software designers can motivate people to reduce their energy consumption by facilitating Cultures of Participation and Meta-Design.

2 Background

While there are many potentially effective ways of causing people to use energy more efficiently [2, 3], I am, for my PhD research, focusing on two ways, namely Psychological Ownership and Cultures of Participation. Both have in common that they facilitate intrinsic rather than extrinsic motivation and that they are currently being under-used in the energy domain.

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 367–370, 2011.

2.1 Psychological Ownership

Psychological ownership [4] describes a state in which a person feels closely connected to an object or idea, to the state that it becomes part of an 'extended self'. As soon as people see something as their own, they value it higher and are more likely to invest time and effort in it [5, 6].

Pierce et al, in a meta-review of research on psychological ownership [4], have identified several factors as requirements for psychological ownership, namely Control, Investment of Self, Intimate Knowing, and Modifiable Targets. If, a target, that is, an object or an idea, fulfills all these requirements, people are more likely to feel ownership for this target.

One reason that too few people are motivated to analyze and improve their energy consumption behaviors is that few are feeling ownership for their own energy consumption or the energy domain. Given the current design of energy systems, this is not surprising. Although the technological infrastructure would allow for it, consumers are **not** giving an option to **control** the price of their energy. The technical implementation and the utility companies do **not** foster or reward **investment of self**.

The electrical grid is designed as a system for passive consumers. Consumers are still being given the same monthly bills that list overall consumption in the abstract unit kWh, **not** supporting **intimate knowing** of how energy is being used or how energy could be saved. And finally, the only thing that consumers can change in the current smart grid is which device they use and how often they use it; the system does **not** provide any **modifiability** to end-users.

2.2 Cultures of Participation

The effects of supportive social environments in which people do not act as solitary individuals but as an active part of an environment are not new; they have been shown to be effective in making people laugh [7], making people contribute and share more [8, 9], making people err on easy questions [10] and even getting people pregnant [11]. But aside from small experimental settings [12, 13], they are not being used to motivate people to use energy more efficiently.

In the current energy system, energy consumption is completely individualistic and invisible to others. Short of choosing to drive a Toyota Prius as a symbol of energy-efficiency or installing solar cells to show interest in renewable energies, people have no way to share their energy attitudes or behaviors. People looking for guidelines or role-models cannot judge or estimate how much more or less energy even their immediate neighbors are using, let alone how their consumption compares to other consumers like them.

3 Approach

3.1 Using Meta-design to Create Psychological Ownership

I am proposing to use the meta-design framework [14] to build a system that fulfill the requirements for psychological ownership. In brief, meta-design is a form of software design that aims to include end-users in the ongoing evolution of the system.

Meta-design environments are supposed to be solution spaces [15] and not solutions. In meta-design environments, users should be able to identify, explore, and re-assess their needs constantly during use time and act as designer that change the environment accordingly when needed.

My system will allow users to understand their energy usage by running and creating energy simulations. These simulations can range from analyzing how much one can use by using a CFL instead of an incandescent light bulb over the time of a year to complex simulations like calculating how switching to a new laundry and different detergent can impact the CO2 emissions over the lifetime of the appliance, allowing **intimate knowing** of their own energy consumption.

Built as a meta-design environment, users can define and use different kinds of devices to be simulated and choose how energy consumption should be represented, for example as CO2 emissions, as monetary savings, or as wheelbarrows of coal that would need to be burned, offering **modifiable targets**. Users can access information from a variety of sources, including the community of other users, and can share their own simulations and insights with others, motivating and rewarding the **investment of self**. By understanding not only where energy is being wasted but what one could do about it, it gives **control** to the users, fulfilling all the requirements for psychological ownership.

3.2 Fostering Cultures of Participation

I am building a social website as part of my software system that addresses the problem of invisible and purely individualistic energy usage by creating awareness of one's own and other people's energy consumption, building a foundation for a Culture of Participation [14] in which more consumers of energy become active and conscious participants of their energy environment. To this end, it is combining awareness tools [16] that keep participants informed about other people's actions and contributions in the online environment with a novel approach to make visible how their own actions influence their environments.

4 Current State

I am reaching a stable state for the theoretical frameworks underlying and guiding the developments. I have created several prototypes for parts of the system and am now implementing the system. I am testing first prototypes with selected groups of interested K-12 students, Amazon Mechanical Turk workers, and local co-workers in the spring of 2011 before deploying a final version to a yet to be determined group later in 2011. I am currently in the process of selecting a suitable test-group for my developments that will make use of my developments for a time-frame of one to three months and that allows me to collect quantitative usage data as well as qualitative data from interviews and observations.

Acknowledgments. I want to thank my co-workers at the Center for LifeLong Learning and Design (L3D) and especially my advisor Gerhard Fischer. My research has been partially supported by the National Science Foundation under Grant No 0968588 and 1028017.

References

1. Intergovernmental Panel on Climate, C.: Climate Change 2007: Synthesis Report. Summary for Policymakers. Intergovernmental Panel on Climate Change (2007)
2. Erhardt-Martinez, K., Donnelly, K.A., Laitner, J.A.S.: Advanced Metering Initiatives and Residential Feedback Programs: A Meta-Review for Household Electricity-Saving Opportunities, vol. E105. American Council for an Energy-Efficient Economy, Washington, D.C., p. 128 (2010)
3. Abrahamse, W., Steg, L., Vlek, C., Rothengatter, T.: A Review of Intervention Studies Aimed at Household Energy Conservation. Journal of Environmental Psychology 25, 273–291 (2005)
4. Pierce, J.L., Kostova, T., Dirks, K.T.: The State of Psychological Ownership: Integrating and Extending a Century of Research. Review of General Psychology (2002)
5. Norton, M.I.: The Ikea Effect: When Labor Leads to Love. Breakthrough Ideas of 2009. Harvard Business Review 87, 30 (2009)
6. Ariely, D.: The Upside of Irrationality. HarperCollins, New York (2010)
7. Cialdini, R.: Influence: Science and Practice, 5th edn. Prentice-Hall, Englewood Cliffs (2008)
8. Cress, U.: Information Exchange with Shared Databases as a Social Dilemma: The Effect of Metaknowledge, Bonus Systems, and Costs. Communication Research 33, 370–390 (2006)
9. Cress, U., Kimmerle, J.: Guidelines and Feedback in Information Exchange: The Impact of Behavioral Anchors and Descriptive Norms in a Social Dilemma. Group Dynamics: Theory, Research, and Practice 11, 42–53 (2007)
10. Asch, S.E.: Opinions and Social Pressure. Scientific American 193, 31–35 (1955)
11. Thaler, R., Sunstein, C.: Nudge: Improving Decisions About Health, Wealth and Happiness. Penguin (2008)
12. Schultz, P.W., Nolan, J.M., Cialdini, R.B., Goldstein, N.J., Griskevicius, V.: The Constructive, Destructive, and Reconstructive Power of Social Norms. Psychological Science: a Journal of the American Psychological Society / APS 18, 429–434 (2007)
13. Schultz, W., Khazian, A., Zaleski, A.: Using Normative Social Influence to Promote Conservation among Hotel Guests. Social Influence 3, 4–23 (2008)
14. Fischer, G.: End-User Development and Meta-Design: Foundations for Cultures of Participation. In: Pipek, V., Rossen, M.B., deRuyter, B., Wulf, V. (eds.) End-User Development, pp. 3–14. Springer, Heidelberg (2009)
15. Fischer, G., Giaccardi, E.: Meta-Design: A Framework for the Future of End User Development. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development, pp. 427–457. Kluwer Academic Publishers, Dordrecht (2006)
16. Kimmerle, J., Cress, U.: Visualization of Group Members' Participation: How Information-Presentation Formats Support Information Exchange. Social Science Computer Review 27, 243–261 (2009)

Beyond Upload and Download: Enabling Game Design 2.0

Navid Ahmadi

Faculty of Informatics, University of Lugano
Via G. Buffi 13, CH-6904 Lugano, Switzerland
navid.ahmadi@usi.ch

Abstract. The participative culture of Web 2.0 has increased the interest of online users in developing interactive artifacts such as games. However, educational computer games as the media to teach computer programming to end users are not cultivated in this culture yet. While several platforms enable sharing games through the Web, the game design process and its educational values are either lost or limited to downloading and uploading source in order to explore and modify the game program. In this research I present AgentWeb, a first-of-a-kind Web-based game design and programming environment. Targeted for the masses, AgentWeb provides visual programming language and runtime system for developing games inside the browser. Built using open Web technologies, AgentWeb can be easily incorporated into the online social networking environments, enabling users to develop, share, explore and customize as they play the game.

Keywords: Social Game Design, Web 2.0, End-User Programming, HTML5.

1 Problem Definition

The participative culture of Web 2.0 [6] has increased the interest of online users in developing interactive artifacts such as games. The pervasive user-created content on the Web, such as wikis and blogs, is the result of shifting online users from passive consumers to active developers. However, the development potentials of online users are suppressed by the static nature of hypertext, the format in which the user-created content is represented. Interactive games are mostly developed by professional developers using Rich Internet Application frameworks such as Flash or Java and embedded into the Web browser. Online users remain passive game players rather than becoming active game developers.

Computer games are particularly used as educational media to introduce school and college students to computer programming [5]. While platforms such as Scratch [8] have enabled sharing educational games through an online social interface, in order to explore and modify the game objects and its program the users have to download them and upload them back into the Website again. On the Website, the game is treated as a black box that users can only play. Respectively, the socialization, e.g., user comments, takes place around the game

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 371–374, 2011.

itself not the game design process. As a result, the existing online social media around end-user developed games fall short of transferring game design and programming knowledge to the observers. Eventually, these Websites turn into an easy-to-access game repository rather than directly mediating the game programming knowledge.

2 Solution Approach

To incorporate the game design fully into the social Web context, the game design environment has to let users explore the game objects and programs right inside their online social environment, without the upload and download barrier. Accordingly, a Web-based game design environment is required to fulfill this fundamental requirement. To maintain the educational values of developed games, the game design environment has to ensure that while exploring the games, the users can socialize over the game objects and program by means of existing Web 2.0 interfaces such as sharing, commenting, rating, and tagging. In other words, users have to be able to inspect the game and its program as a white box software rather than a black box software that is only being played.

Two main challenges to build a Web-based game design environment are supporting novice programmers and the development of such a system using Web technologies. The focus of this research is to deliver the educational values of game design to the masses who typically do not have programming skills. Accordingly, the environment should provide users with a low-threshold end-user programming language. This challenge can be addressed using existing techniques such as visual languages [2] and domain-oriented design environments [3]. At the implementation level, the Web technologies become challenging as they were not initially intended to support such a dynamic environment in which the users program the games. Only in one decade and mostly motivated by the fast growth of the Web users and emergence of new applications the Web technologies have evolved to address many of drawbacks [4]. Poor JavaScript performance and lack of support for graphics rendering have been addressed in recent JavaScript engines and emerging HTML5 features. Yet, developing a game design environment for the masses, in which the user interface is a prominent matter, is highly challenging due to the lack of high-level APIs even for the simplest user interface features such as drag-and-drop. Respectively, many basic features should be implemented from scratch using JavaScript as the *assembly* language of the Web in order to build a visual programming language and game engine runtime.

3 Development

I have designed and implemented AgentWeb, a first-of-a-kind 100% Web-native game design environment. Users design and program their games right inside the Web browser. Built upon only open Web technologies, all the game objects including game characters, their programs, and the game scenes are stored as JavaScript and HTML5 objects. Therefore, not only users play the games, but also they can explore and modify all the game objects including the program of game characters from inside the browser. The barrier of downloading and uploading the games do not exist anymore.

AgentWeb's game design and programming model resembles AgentSheets [7], a desktop-based game authoring tool used by novice programmers. Game objects are called agents. Users create the agents, depict them using the provided image editor, program them using a visual programming language, and draw the agents in the game scene. Figure 1, shows the AgentWeb game design environment used for developing a Frogger game.

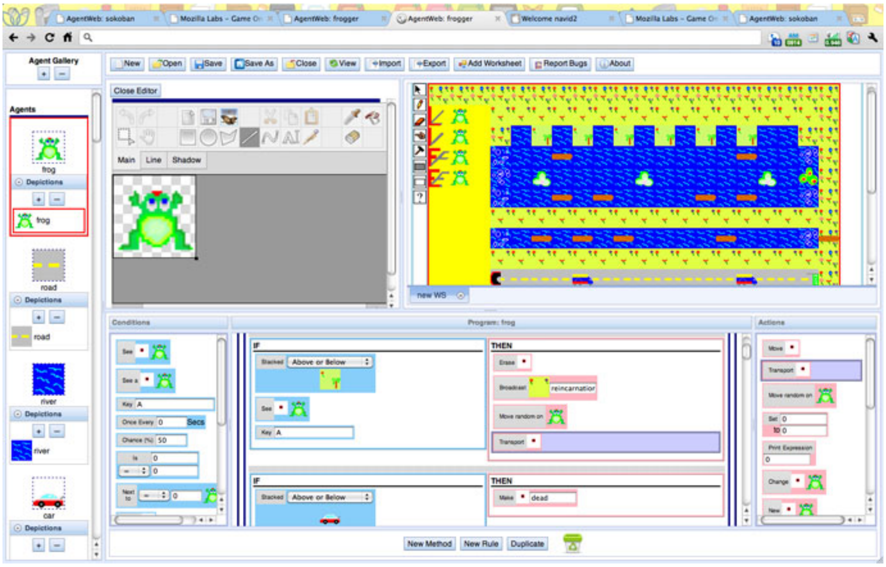


Fig. 1. AgentWeb: a Web-based game design environment targeted for online users

AgentWeb consists of a visual programming environment in order to lower the programming barrier to end users. Moreover, the visual programming environment enables rapid development of sophisticated video games to a point which a game as sophisticated as Frogger is built by a novice programmer in 60-90 minutes. Usability studies conducted with middle school students shows that students create interactive games using AgentWeb at the same pace as desktop-based AgentSheets.

The programming language relies on a compiler which dynamically translates the end-user developed visual program into JavaScript at runtime. Benefiting from dynamic compilation, AgentWeb supports *interactive programming* [1] using which users are capable of modifying the agent program while the game is running and the changes are immediately applied into the execution. Interactive programming lets users explore the behavior of agents and receive immediate feedback while they programs. The performance evaluation of execution and rendering the end-user developed games inside the Web browser reveals a high performance comparable to desktop-based equivalent implementations, and in some cases even outperforming them.

4 Conclusion and Future Work

While online users are interested in developing more sophisticated artifacts such as games, existing desktop-based end-user game design environments can not be fully integrated into and benefit from the participative culture of Web 2.0. As a result, the social and educational values of end-user developed games are omitted. In this research, I present a Web-based game design environment, called AgentWeb. Built using open Web technologies AgentWeb can seamlessly integrate into social Web environments, enabling users to not only share their games, but also explore, modify, and share the game objects and program with no upload and download barrier. The future work includes integrating AgentWeb into existing social networking environments while developing social interfaces through which online users socialize around the game design process.

Acknowledgments

This research has been supported in part by the Swiss National Science Foundation. Sincere thanks go to professors Mehdi Jazayeri, Alexander Repenning, and Gerhard Fischer and Dr. Francesco Lelli for their deep involvement and support.

References

1. Barstow, D.R., Shrobe, H.E., Sandewall, E.: Interactive programming environments, p. 609. McGraw-Hill, New York (1984)
2. Burnett, M., McIntyre, D.: Visual programming. Computer 28(3), 14–16 (1995)
3. Fischer, G.: Domain-oriented design environments. Automated Software Engineering 1(2), 177–203 (1994)
4. Jazayeri, M.: Some trends in Web application development. In: 2007 Future of Software Engineering (FOSE 2007), pp. 199–213 (May 2007)
5. Lewis, C., Repenning, A.: Creating educational gamelets. In: DiGiano, C., Goldman, S., Chorost, M. (eds.) Educating Learning Technology Designers: Guiding and Inspiring Creators of Innovative Educational Tools, pp. 203–229. Routledge, New York (2008)
6. O'Reilly, T.: What is Web 2.0 - design patterns and business models for the next generation of software (2005), http://oreilly.com/web2/archive/what-is-web-20.html
7. Repenning, A., Ambach, J.: Tactile programming: a unified manipulation paradigm supporting program comprehension, composition and sharing. In: Proceedings of the 1996 IEEE Symposium on Visual Languages, pp. 102–109 (August 1996)
8. Resnick, M., Silverman, B., Kafai, Y., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J.: Scratch: programming for all. Commun. ACM 52(11), 60–67 (2009)

Infrastructuring with a Focus on Developing End-Users Capabilities

Johan Bolmsten[1,2]

[1] IT University of Copenhagen, Rued Langaards Vej 7, 2300 Copenhagen S, Denmark
[2] World Maritime University, Citadellsvägen 29, 21118 Malmö, Sweden
jb@itu.dk

Abstract. In an ongoing PhD study, a shop floor model of development is developed in terms of approaching Participatory Design (PD) as an organizational implementation strategy. The focus is how domain experts continuously can exercise influence over development given increasingly complex circles of technical and organizational software system "infrastructuring".

Keywords: End-User Development, ICT-Infrastructure Development.

1 Introduction

The World Maritime University (WMU) in Malmö, Sweden, pursues a development model derived from the shop floor [1, 2] for the in-house design of software to support the daily work activities. The empirical case underpinning an ongoing PhD study starts with the growing opportunities and challenges of developing and integrating software infrastructure support beyond individual application areas and departmental boundaries. A shop floor model of development is in this study developed in terms of approaching Participatory Design (PD) as an organizational implementation strategy. The focus is how domain experts continuously can exercise influence over development given increasingly complex circles of technical and organizational software system infrastructuring.

2 Infrastructuring "in the Wild"

PD has a mature tradition of targetting diversity in design of software support. For this study, Dittrich et al's [2] "in the wild" perspective serves as a useful lens that places the continuous design and use practices that domain experts and software engineers use in the centre. The objective of research intervention thereby is not so much to introduce a PD way of doing things from outside, as it is to work with the evolvement of software support from the realities of everyday work – on the shop floor [1]. A for this study useful extension of the "in the wild" perspective is the process of "infrastructuring" [3]. What "infrastructuring" from an "in the wild" perspective supports is essentially to discuss how Suchman's [4] notion of "artful integration" takes place; that is the evolution of software support "design[ed] from somewhere" as

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 375–378, 2011.

opposed to the production of discrete devices “design[ed] from nowhere”. The contributions of all the papers in this study build on the interaction between technology and participatory sustainable evolution with respect to heterogeneous and developing requirements in iterative cycles of increasingly complex technical and organizational “infrastructuring”. Each interaction enables both an improved understanding and ability for continuous deliberations of better software support to improve shop floor work practices. The result is an account of dimensions of design and use that would not have been visible within an isolated local project intervention on its own.

3 Research Approach

Early in the research process the notion of “very action research” was coined to refer to the methodological approach taken: I am employed by WMU as a software engineer doing technical programming and working to bring use and design together on a local project and organizational arena level. My professional work at WMU is complemented with action research as a PhD student for the IT-University of Copenhagen. The main empirical base is constituted by my day-to-day interactions with faculty, staff, and students in the design of useful software support for their work.

The ethnographically inspired Cooperative Method Development (CMD) action research cycle put forward by Dittrich, Rönkkö, Eriksson, Hansson, & Lindeberg [6] is used as a methodological framework. The CMD research cycle focus on shop floor development practices where the practitioners perspectives are the focal point when evaluating empirical research and deliberating improvements. This is a process that is done together with involved practitioners. The action research cycle consists of three phases: (1) *Understanding practice; (2) Deliberation of improvements; (3) Implement and observe improvements.*

4 A Long-Term Case Study of Infrastructure Development

The nature of the action research conducted has enabled a long-term and continuous intervention in software development practices and strategies relating to matters of working with PD as an organizational implementation strategy. This has made it possible to account for a range of dimensions of relevance if pursuing an organizational viable PD approach to software development - within and beyond the local project context. The study, so far, covers three papers that are chronologically situated in the intervention and reflection that is made possible within the evolving realm of my position as an IT-professional at WMU. Although the papers overlap, they are targeted to focus on three different core areas of relevance to software engineering [7, 8]: Technology (Paper 1); Process (Paper 2); Use (Paper 3).

4.1 Technology Matters – The Role of the Technical Base in Infrastructure Development and Evolution.

Software is being used more and more to support cross-organizational collaboration providing an infrastructure for cooperative practices. Different items of functionality developed in local projects at the same time need to be integrated in an infrastructure and share a common technical base. This presents new challenges for design and development processes as well as the technical base for the evolving infrastructure. Participatory Design promotes the involvement of users into IT development processes. Web 2.0 is advocated as a technical base that provides the flexible integration of heterogeneous applications and own contents and functionality and even opens up development possibilities for end users. However Web 2.0 can have different traits. In the article we report on the evolution of a technical base for the IT infrastructure of WMU. The specific characteristics of the technical base turned out to influence not only the possibilities of integrating different applications but also the cooperation of professional developers and end users, and the design space for end-users as well as developers. In this sense, the choice of the technical base thus came to both enable and constrain the possibilities for participatory design. These results to contribute to the understanding and tackling of the challenges long-term collaboration and co-construction of IT infrastructures provide for Participatory Design and Information Systems Development.

4.2 Who's Afraid of the Big Bad Wolf? - Leveraging the Capabilities of Shop Floor IT Design on the Organizational IT Management Arena

Organizational IT management is an area that has been approached with caution in contemporary Participatory Design (PD) research. This study shows how and why PD can stop to be afraid of what sometimes seems to be treated as the big bad wolf!

The study is based on action research where one of the authors is employed at the World Maritime University (WMU) in Malmö, Sweden, as an in-house IT-professional. During the course of six years, he has worked together with his fellow colleagues to understand and deliberate on how a better organizational IT management can be accomplished. Based on the empirical research, the article contributes to the understanding of how local shop floor development can take place in an increasingly complex technical and organizational infrastructure environment; and how key domain users and IT professionals can be influential in transforming this model of development to an organizational implementation strategy.

4.3 Infrastructuring When You Don't – End-User Development and Organizational Infrastructure

Technologies promoting End-User Development enable domain experts to adjust and develop tools to fit with their specific work practice and thus to be efficient with respect to their professional tasks. In today's organizations, however, single applications become part of organizational infrastructures. Such infrastructures enable integration between different applications and tasks but, at the same time, introduce new constraints to ensure interoperability. How can the advantages of End-User Development be kept without jeopardizing the advantages of integration between different applications? The third article presents an empirical study on End-User Development in the context of the development of an organizational ICT infrastructure. Based on the analysis of the empirical material we discuss the challenges the infrastructure context provides for End-User Development.

5 Concluding Remarks and Future Research

A second research cycle will continue to target design and use in a complex infrastructure environment to add in an innovatory way of how software engineers and domain experts can cooperate around the development of software support. In a landscape of additionally complex technologies and a larger number of domain experts participating in an Enterprise Resource Planning system, the challenge is to be able to continuously nurture a locally anchored development style that gains its momentum from the domain experts themselves.

References

1. Eriksén, S.: Knowing and the art of IT management: an inquiry into work practices in one-stop shops (1998)
2. Dittrich, Y., Eriksén, S., Hansson, C.: PD in the Wild; Evolving practices of Design in Use. In: Proceedings of the Participatory Design Conference (2002)
3. Star, S.L., Bowker, B.G.C.: How to Infrastructure. In: Lievrouw, L.A., Livingstone, S.M. (eds.) Handbook of New Media: Social Shaping and Consequences of ICTs, p. xxiv, 564. SAGE, London (2002)
4. Suchman, L.: Working relations of technology production and use. Computer Supported Cooperative Work 2(1), 21–39 (1994)
5. Hanseth, O., Braa, K.: Hunting for the treasure at the end of the rainbow: standardizing corporate IT infrastructure. Computer Supported Cooperative Work 10(3), 261–292 (2001)
6. Dittrich, Y., et al.: Cooperative method development. Empir. Software Eng. 13(3), 231–260 (2008)
7. Floyd, C.: Software Development as Reality Construction. Software Development and Reality Construction (1992)
8. Floyd, C., Reisin, F., Schmidt, G.: STEPS to software development with users. ESEC 89, 48–64 (1989)

BioShape: End-User Development for Simulating Biological Systems

Federico Buti, Diletta Cacciagrano, Massimo Callisto De Donato,
Flavio Corradini, Emanuela Merelli, and Luca Tesei

University of Camerino, School of Science and Technology
Via Madonna delle Carceri 9, 62032, Italy
name.surname@unicam.it
http://www.cs.unicam.it

Abstract. The simulation and visualization of biological system models are becoming more and more important, in both clinical and research activities. Many tools help biologists and bioengineers to analyse and to study complex biological phenomena, such as disease spreading, tissue development and neurological reactivity.

We present ongoing work on BioShape, a bio-inspired 3D simulation tool whose novelty consists of providing a 3D geometry-oriented modelling environment. Unlike most of the other tools, BioShape development aims to improve usability by taking advantage of End-User Development techniques. While the user can easily understand the basic features of the tool, he is also made capable of extending them at different levels of complexity, for specific simulation purposes.

Keywords: Systems biology, Particle-based models, Simulation of biological systems, Simulation tool.

1 Introduction

Nowadays, systems biology plays an important role in the study and in the development of new improved medical solutions. In silico simulation and prediction of pathological phenomena [1], tailoring of medical treatments based on the characteristics of an individual patient ([2], [3]) as well as the research of nanotechnology applications are only some examples. In silico experimentation, in particular, permits to execute tests that would be critically expensive if carried out through *in vitro*/*in vivo* techniques - e.g. vascular blood diffusion in heart diseases, signal passing in neurological tissue and so on. Most notably, it allows to study phenomena or interactions that do not occur, or occur rarely, in nature.

Since biological processes are inherently complex and computationally expensive, simulation tools often concentrate on the improvement of computational efficiency to gain better outcomes in lesser time. As a result, other important aspects such as the *Human Computer Interaction* (HCI) and the usability are overshadowed and most of these tools tend to push their grain over the user [4].

Recently, we proposed BioShape [5], a bio-inspired simulation tool with a 3D geometry-oriented modelling environment. Unlike other bio-tools, BioShape

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 379–382, 2011.

allows biologist and bio-engineers to represent biological processes by taking into account properties such as: *proximity*, *spacial separation*, *migration*, *molecular diffusion* and so on. In this work we demonstrate how End-User Development approaches (EUD) [6] can be successfully applied to BioShape in order to improve its usability and flexibility.

The paper is organised as follows: Section 2 provides a general description of the tool. Section 3 shows our strategies to simplify and to improve user application tasks. Section 4 summarises and traces ongoing and future work.

2 BioShape

BioShape[1] is a spatial 3D simulator that provides a particle-based, geometry-oriented modelling environment. Every simulated element can be treated as a particle, *an entity*, with a specific shape, perception capabilities and a personalised motion law. Entities move and interact into the simulated space, *the simulation environment*; a two-phase collision detection algorithm [7] guarantees that colliding entities that cannot bind will repulse each other according to their motion laws.

Small elements which do not interfere with the collision detection, but affect the simulated process, are coded as *grouped entities*, i.e. specific gradients over the simulation environment. Ions or water molecules are two common examples.

BioShape accepts as input *XML datasets*. A typical dataset contains all the necessary information about the entities (e.g. type, position, velocity) and the environment (e.g. dimensions, boundaries, defined grouped entities). We refer the reader to [5] for a complete description of BioShape's architecture.

3 EUD: Simulation Power to the User

In this section we describe the ideas that are currently under development in BioShape. Following the "*gentle slope*" approach ([8], [6]), the tool will provide the user with three levels of EUD activities of increasing complexity.

First level - customization. At this level the user with a little knowledge of the tool can create her/his own basic simulation. Two main aspects are defined: the simulation environment and the entities which move and interact in the simulated space. The default environments expose basic behaviours such as: finite bordering (the simulation is *closed* in the simulated space), purity/semi-purity (default sets of sliders for grouped entities - water, potassium and so on - or no grouped entities at all, depending on the selected default environment). The user is provided with a set of default entities with basic shapes, mainly cones, cylinders, cubes and prisms. They come with some degree of control over basic aspects such as velocity, speed rotation, etc. Both environments and entities can be *annotated* so that the user can incorporate domain-specific information. For

[1] The BioShape Project - http://cosy.cs.unicam.it/bioshape/

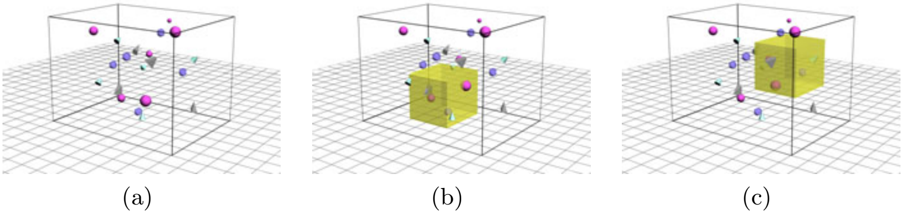


Fig. 1. (a) Simulation environment, (b) enriched with a single diffusion box, (c) displacement of the diffusion box

instance, a simple sphere entity can be annotated to describe it as a metabolite as well as a cell of a tissue.

Second level - integration. The user can *glue* together default entities to create new complex ones not available in the default library. By defining a series of anchor points or anchor surfaces on the component entities, he can create *ad hoc* entities for its own simulations. Sliders and annotations are again used to specify the properties about the new defined complex entities. Same strategies can be applied to environments, to glue them together. "*Diffusion boxes*" are used to intuitively add grouped entities to the environment through coloured box. Boxes can be dragged in order to define the diffusion area while the density is described by the *intensity* of the colour. At any time, the user has a direct outlook of all the grouped entities and of their position in the space (see figure 1).

Third level - extension. At this level the user can generate its own environments and entities in order to create a very specific simulation. The entities creation phase can be carried out through the "inflatable icons" approach [9]: the user can draw a simple 2D form and then *inflate* it in order to create a 3D form (see figure 2). Inflation can be applied with different strengths to different portions of the 2D/3D entity to create particularly complex entities. Also simulation environments can be generated through inflation. Moreover, the environments can be further characterised by defining specific behaviours of the walls w.r.t. contained entities. For instance, the user can specify that a type of

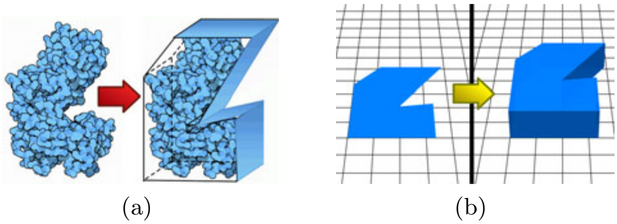


Fig. 2. (a) Hexokinase enzyme approximation to an entity and (b) Hexokinase generation through inflation

entity can pass through a wall and *disappear* from the simulation (the so called "open boundaries") or *reappear* on another wall. This flexibility allows to simulate particular case studies such as the section of a vein, in which a simulation wall "produces" entities (e.g. erythrocytes) and another one "dissolves" them.

4 Conclusions and Future Work

In this short paper, we have presented the ongoing work on BioShape tool and we have shown how EUD techniques can be successfully applied to improve important aspects such as usability and flexibility.

The gentle slope technique guarantees that the user is not overcome with tool grain. Instead, it provides the user with the right means to gradually understand tool potentialities and to exploit them. Users can easily specify models with a high degree of personalisation (e.g. how entities interact, collide with other entities or with the defined boundaries) or they can create their own simulation assets. As a side effect, developers are set free from implementing all the possible simulation scenarios which would be difficult if not impossible.

Currently, a first version of the tool is under development. It includes some basic components - a cubic simulation environment, the distributed collision detection algorithm, the node coordination logic for distributed computation - and the customization level presented. The second and third levels are also scheduled and we plan to implement them in the next releases of BioShape.

References

1. Grupe, A., Germer, S., Usuka, J., Aud, D., Belknap, J.K., Klein, R.F., Ahluwalia, M.K., Higuchi, R., Peltz, G.: In silico mapping of complex disease-related traits in mice. Science 292, 1915–1918 (2001)
2. Manos, S., Zasada, S., Coveney, P.V.: Life or death decision-making: the medical case for large-scale, on-demand grid computing. CTWatch Quarterly 4, 1–8 (2008)
3. Taylor, C.A., Draney, M.T., Ku, J.P., Parker, D., Steele, B.N., Wang, K., Zarins, C.K.: Biomedical paper predictive medicine: Computational techniques in therapeutic decision-making. Computer Aided Surgery 4, 231–247 (1999)
4. Dix, A., Finlay, J.E., Abowd, G.D., Beale, R.: Human-Computer Interaction, 3rd edn. Prentice-Hall, Englewood Cliffs (2003)
5. Buti, F., Cacciagrano, D., Corradini, F., Merelli, E., Tesei, L.: Bioshape: a spatial shape-based scale-independent simulation environment for biological systems. Procedia CS 1, 827–835 (2010)
6. Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-User Development: An Emerging Paradigm. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development. Human-Computer Interaction Series, vol. 9, pp. 1–8. Springer, Netherlands (2006)
7. Mirtich, B.: V-clip: Fast and robust polyhedral collision detection. ACM Trans. Graph. 17, 177–208 (1998)
8. Mørch, A.: Three Levels of End-User Tailoring: Customization, Integration, and Extension, pp. 51–76. MIT Press, Cambridge (1997)
9. Repenning, A.: Inflatable icons: Diffusion-based interactive extrusion of 2d images into 3d models. J. Graphics Tools 10, 1–15 (2005)

Supporting End-User Development of Web Sites through MAMBA

Nicola Gelfi

Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia,
Via Branze 38, Brescia, 25123, Italy
nicola.gelfi@ing.unibs.it

Abstract. This paper presents EUD-Mamba, a prototype system for End-User Development of web sites. EUD-Mamba's goal is to allow users to build a web site simply by describing the content they want to publish in a conceptual way: what the content is and how it should look like in the site. The system is based on the MAMBA (Multi-device Adaptive Model-Based Approach) web personalization engine. The proposed approach is substantiated by a running example showing how to build a simple PhD personal web site using EUD-Mamba.

Keywords: End-User Development, web development tools, web site design, web personalization.

1 Introduction and Related Work

This paper presents EUD-Mamba, a prototype system for End-User Development of web sites, that relies on the MAMBA (Multi-device Adaptive Model-Based Approach) web personalization engine, which implements a novel computational model for dynamically adapting the information to be presented to the user according to a set of user interface design principles [1].

EUD-Mamba's ultimate goal is supporting a novel design methodology, where the developer of a web site is completely free from the task of presentation design and can concentrate on the production of content. Differently from a traditional content management system, which requires the user to select from a library the preferred layout and theme and then to insert the desired content in it, it is possible to specify only the content and the individual presentation preferences, leaving the choice of the most suitable presentation to the system.

Our approach follows the idea of model-based development, in which the users just provides a conceptual, high-level description of the content and the system generates the corresponding web site [2].

In EUD-Mamba, the end user's participation is entirely focused on the initial design phase [3] of the web site, rather than during the usage of the site itself. Better capturing and satisfying user requirements, which is at the center of EUD [5], means in EUD-Mamba context to allow the user to describe the content he/she wants to publish and let the system find the best presentation for the content, as well as the navigation path between the site's pages. EUD-Mamba's wizards offer an important

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 383–386, 2011.

mean to bridge the "communication gap" between the technical view of professional web site designers and the domain expert view of end users [2] [4].

Research in End-User Web Development (EUDWeb) has focused mainly on the development of web applications (e.g. FAR [7], DENIM [8] or WebSheets [9]), rather than web pages that simply present information, which, by the way, represent the vast majority of the web pages developed by end users [6].

An analysis of state-of-art web development tools conducted in [6] has pointed out that most commercial web development tools for nonprogrammer developers are still in their infancy. Even for creating a basic web site the user has to cope in some way with programming languages details, such as HTML. In our approach, the user can provide the content in a completely conceptual way: what the content is and how it should look like in the site in an intuitive and informal way, and the system takes entirely care of the site generation.

2 Overall Approach and System Architecture

Our approach in based on three fundamental elements, called *content assignment, presentation schemes* and *user interface design principles.*

By answering EUD-Mamba's questions, the users store the content to be displayed on the web site and its description in a tree-based structure called *content assignment.*

This structure is passed on to the MAMBA engine, which takes care of the selection of the best presentation templates to be used for the site generation in a library of available templates, called *presentation schemes*, that specify a large variety of possible presentations suitable for a wide class of exigencies, users, and devices. This library is built by external designers and is part of the system knowledge base.

The selection of which presentation schemes are the most suitable for the supplied content assignment is carried out by the matching and ranking process, which takes into account a collection of general *user interface design principles* that embody the best practices in the area of user interface design [1]. This process is embodied within the MAMBA engine, as well as the module which takes care of the actual HTML page generation and linking.

3 Building My PhD Web Site with EUD-Mamba

In this section I will briefly show the construction of a sample web site containing my personal information. Assume that I want to publish on my web site a picture containing my portrait, my curriculum vitae (title and text) and my publications (titles and DOI links). Upon the insertion of each piece of content, EUD-Mamba first asks what type the content belongs to (currently supported types are image, text or link). Then it asks a small number of specific questions concerning its individual presentation preferences, depending on the type of the content (see Fig. 1(a)).

First, the user is asked to describe the basic semantics of the content through a keyword called *category identifier.* This will aid the system to find the most suitable place for the content in the final web site. Note that the choice can be completely left to the system, as the *category identifier* is optional. The second questions asks the user to state if the provided content has to be displayed at all costs in the site, or if the system may choose to discard it in order to find a better presentation for the rest of the site's content. Finally the user has to tell the system what grade of emphasis the piece of content he/she is providing should have when displayed in the site. The higher the grade of emphasis specified, the more the system will try to display the content in a very visible position of the final web page.

For example, I have described my portrait's basic semantics to the system as a “portrait”; I have also told the system that my portrait is absolutely necessary for the site, and that is should be highly emphasized in the presentation. As an additional feature to guide the site generation, the user can specify groups of related content as a *composite* piece of content (e.g. in this case, my publications, see the “Publications” item in Fig. 1 (b)). The system will try to display composite content in constrained areas of the web site, such as frames, boxes or numbered lists. After clicking on the “Generate Website” button, the generated *content assignment* in sent to the MAMBA engine and the generated two-page site is shown in Fig. 2. Note that the “Next” and “Previous” links that connect the two pages are not present in the original presentation schemes: the MAMBA engine has automatically segmented the *content assignment* into two pages and has created the navigation path that connects them.

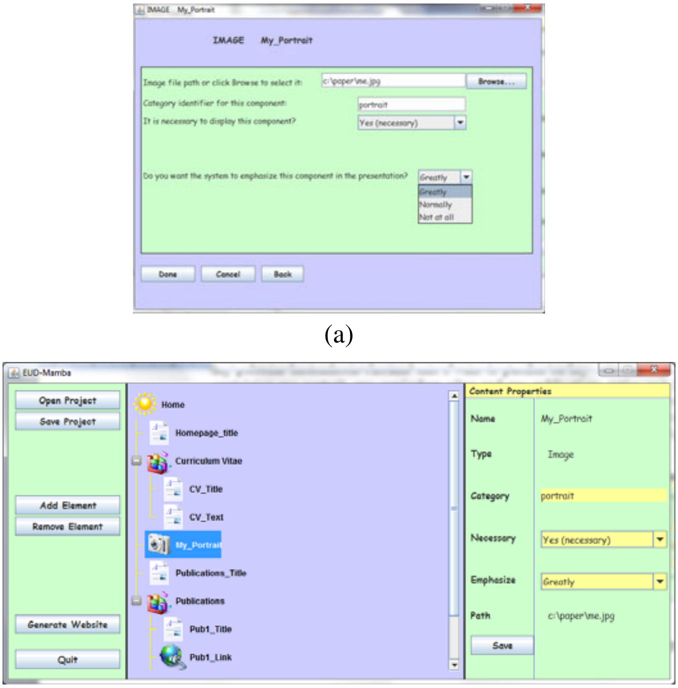


IMAGE My_Portrait

IMAGE My_Portrait

Image file path or click Browse to select it: c:\paper\me.jpg

Category identifier for this component: portrait

It is necessary to display this component? Yes (necessary)

Do you want the system to emphasize this component in the presentation? Greatly

(a)

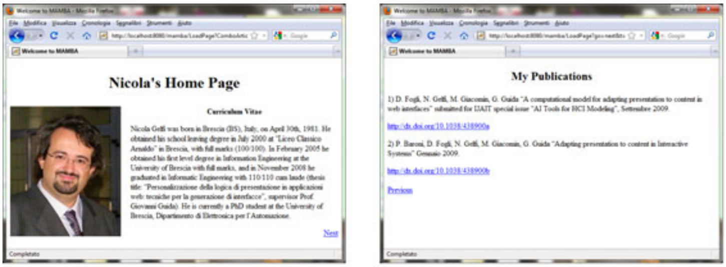
EUD-Mamba

Content Properties

Name	My_Portrait
Type	Image
Category	portrait
Necessary	Yes (necessary)
Emphasize	Greatly
Path	c:\paper\me.jpg

(b)

Fig. 1. (a) The wizard appearing upon the insertion of the picture containing my portrait and (b) EUD-Mamba main screen showing all the content of the sample web site


Welcome to MAMBA - Mozilla Firefox
File Modifica Visualizza Cronologia Segnalibri Strumenti Aiuto
http://localhost:8080/mamba/LoadPage?ComboArtic
Google
Welcome to MAMBA
Completato
Welcome to MAMBA - Mozilla Firefox
File Modifica Visualizza Cronologia Segnalibri Strumenti Aiuto
http://localhost:8080/mamba/LoadPage?go=nextSits
Google
Welcome to MAMBA
Completato

Nicola's Home Page

Curriculum Vitae

Nicola Gelfi was born in Brescia (BS), Italy, on April 30th, 1981. He obtained his school leaving degree in July 2000 at "Liceo Classico Arnaldo" in Brescia, with full marks (100/100). In February 2005 he obtained his first level degree in Information Engineering at the University of Brescia with full marks, and in November 2008 he graduated in Informatic Engineering with 110/110 cum laude (thesis title: "Personalizzazione della logica di presentazione in applicazioni web: tecniche per la generazione di interfacce", supervisor Prof. Giovanni Guida). He is currently a PhD student at the University of Brescia, Dipartimento di Elettronica per l'Automazione.

Next

My Publications

1) D. Fogli, N. Gelfi, M. Giacomin, G. Guida "A computational model for adapting presentation to content in web interfaces" submitted for IJAIT special issue "AI Tools for HCI Modeling", Settembre 2009.

http://dx.doi.org/10.1038/438900a

2) P. Baroni, D. Fogli, N. Gelfi, M. Giacomin, G. Guida "Adapting presentation to content in Interactive Systems" Gennaio 2009.

http://dx.doi.org/10.1038/438900b

Previous

Fig. 2. The two-page web site generated by MAMBA from the content I provided

4 Future Work

The present work opens up several directions for further research. Extending the computational model here proposed to manage dynamic web pages and web applications can greatly extend the potential of EUD-MAMBA and open up new application opportunities according to the well-known paradigm. Furthermore, connecting EUD-MAMBA to an ontology engine and introducing new UI design principles that take into account on the semantic relations between the content items could overcome the limitations of the current *category* identifiers and might greatly contribute to the quality of the generated web site.

References

1. Fogli, D., Gelfi, N., Giacomin, G., Guida, G.: A computational model for adapting presentation to content in web interfaces. International Journal on Artificial Intelligence Tools 19(6), 783–818 (2010)
2. Paternò, F.: Model-based design and Evaluation of Interactive Applications. Springer, London (2001)
3. Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-User Development: An Emerging Paradigm. In: Lieberman, H., et al. (eds.) End User Development, pp. 1–8. Springer, Heidelberg (2006)
4. Majhew, D.J.: Principles and Guideline in Software User Interface Design. Prentice-Hall, New York (1992)
5. Stiemerling, O., Kahler, H., Wulf, V.: How to make software softer – designing tailorable applications. In: Proceedings of the ACM Symposium on Designing Interactive Systems (DIS 1997), August 18-20, pp. 365–376. ACM Press, New York (1997)
6. Rode, J., Rosson, M.B., Perez Quinones, M.A.: End User Development of Web Applications. In: Lieberman, H., et al. (eds.) End User Development, pp. 161–182 (2006)
7. Ambler, A., Leopold, J.: Public Programming in a Web World. In: IEEE Symposium on Visual Languages, Nova Scotia, Canada, pp. 100–107 (1998)
8. Burnett, M., Chekka, S.K., Pandey, R.: FAR: And End user Language to Support Cottage E-Services. In: HCC-2001 IEEE Symposiam on Human-Centric Computing Languages and Environments, Stresa, Italy, pp. 195–202 (2001)
9. Wolber, D., Su, Y., Chiang, Y.T.: Designing Dynamic Web Pages and Persistence in the WYSIWIG Interface. In: IUI 2001, San Francisco, CA, USA (2002)

Web of Active Documents from the End-User Perspective

Iade Gesso

Università degli Studi di Milano - Bicocca
Viale Sarca 336, 20126 Milano, Italy
iade.gesso@disco.unimib.it

Abstract. This paper presents the development of two visual editing tools that support the end-users in easily customizing the active documents of the WoAD framework, both in their structure and the rule-based active behaviors, to better support their daily work practices. After summarizing the studies that inspired WoAD, the paper illustrates the expected goals of this work, some implementation details and the expected contributions.

Keywords: WoAD, Active Documents, EUD, Visual Editors.

1 Motivation and Background

Most of organizational working environments involve a considerable number of workers that are usually organized into relatively small working groups and share a common goal to reach. These groups of workers use in their work practices a certain set of documents, that can be seen as a *web of documents*. During their activities, the workers develop and maintain some document-based *conventions* that support their group in reaching its goals (e. g., an annotation on a document could make the workers aware about some relevant contextual information, helping them to accomplish their current task) and in coordinating its members [1]. The conventions are actually a component of the wider set of contents that constitute the *knowledge* that has been developed by the group of workers during their interactions. Thus, through the documents it is also possible to evoke some useful knowledge to help the workers in managing the situations that can occur during their daily work, according to the commonly agreed practices of their working group. The process of creation of this knowledge (including the conventions) is often *informal* and *local* to each group, and it is a stratified and cyclic process[1].

The introduction of organizational information systems, that allow for the adoption of the digital documents in substitution of the conventional paper-based ones, may have a negative impact on this process of creation of knowledge and conventions. This is mainly due to the lack of attention given by system designers to these aspects characterizing the use of the documents by the workers. Thus, providing a really effective support to the local work practices of the

[1] The knowledge creation process follows the spiral model proposed by Nonaka and Takeuchi [8], from the *explicit* to the *tacit* (local and informal) form and vice versa.

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 387–390, 2011.

groups could reduce the negative impacts deriving from the adoption of digital document information systems inside the work settings. In order to reach this goal, a good solution is to preserve as much as possible the possibility to convey to the workers some additional data and information directly through the digital documents, only when it is needed and unobtrusively, as well as this happens with the paper-based documents.

The provision of these information could be made *proactively*, using the concept of *active documents*. One of the most notable research activities about active documents has been the *Placeless Document Project* [6]. Unlike what happens in traditional systems, *placeless documents* are managed through their properties (i. e., metadata) rather than through their location, and some of these properties can carry small pieces of executable code, making the documents able to react to some conditions with some active behaviors.

The information that have to be proactively conveyed to the workers can be grouped under the notions of *Awareness Promoting Information* (API) and *Knowledge Evoking Information* (KEI) [3]. API and KEI can be conveyed through suitable *affordances*[2] and differ in their associated meaning: an API is any additional information that could make the workers aware about some relevant thing (e. g., particular context conditions or events), and a KEI is any additional data that allows the workers to get a direct access to or simply evoke in users' minds some useful knowledge resources.

The above concepts inspired the *Web of Active Documents* (WoAD) framework [5], a *design-oriented* framework that encompasses both a *conceptual model* and a *reference software architecture*. WoAD combines the concepts of *web of documents* and *active documents*, extending the latter using symbolic and declarative expressions to define the executable code that specifies the proactive behaviors. The documents are composed by two intertwined parts: a *passive part*, i. e. the container of the data with its own specific structure, and an *active part* (the *mechanisms*), i. e. the above mentioned executable code that augments the documents contents. To define the mechanisms WoAD encompasses the denotational language LWOAD [4]. Each mechanism is divided in two distinct parts: the *antecedent* (i. e. the 'if' part) that holds the set of conditions to be checked, and the *consequent* (i. e. the 'then' part) that defines the sequence of simple actions to execute to convey the right APIs or KEIs, when the conditions in the antecedent are met.

2 Research Questions

Like in the case of the work practices, also the structure of paper-based documents is influenced by the local habits and needs of each group of workers. Unfortunately, most of the document-based information systems does not allow for changing the structure of their digital documents, that usually are forms with a fixed and rigid structure that is imposed at the design time and forces the workers to follow a fixed sequence of filling in operations.

[2] The functionality related to the hint that the system provides.

Moreover, focussing on the WoAD framework, while it is true that the use of LWOAD allows workers to create their own mechanisms in a flexible and powerful manner, however this is strictly influenced by their skills: the workers are expert of their applicative domain, but they have, for several reasons, a lower confidence with any kind of formalized language, that must comply with an inner logic and strict syntactic constraints.

Thus, the goal to reach is to allow for the WoAD-based information system to be more flexible and "tailorable" with respect to the local needs of each group of workers, leaving them, as actual end-users of the system, as much as possible free to directly customize the documental system by defining both the structure of the documents and the related mechanisms, according to the tenets of the *End-User Development* (EUD) research field [7].

Then, in order to reach this goal, the introduction of some graphical tools that allow the workers both in defining the structure of their documents (*document templates*) and in creating their own mechanisms in a more abstract, even if less flexible and powerful manner than the direct use of the LWOAD language.

3 Plan of Work

In order to meet the aimed goals, a preliminary consolidation of the existing WoAD architecture [2] has been performed, and consists of a *reorganization* of the framework components, including those that will support the new editing environments, and a solution for the *data persistence* that is independent of the underlying storage solution (i. e., DBMS, storage files).

The next step consisted in the identification of an existing easy to use platform, that provides a visual editing environment (e. g., based on the drag and drop), to be used as the basis for the development of the WoAD editing tools. After an analysis of the existing solutions, the chosen platform has been Oryx Editor[3], an open-source, web-based and extensible modelling platform that can be easily customized through the creation of sets of plug-ins. Thus, the present customization implements a document templates editor that allows the end-users (i. e., the workers) to create the structure of a document, and storing it into a repository, with a simple versioning capability.

At the same time, another analysis has been performed to find an existing solution that allows for the visual creation of the mechanisms, and their translation into the JBoss Drools syntax. Also in this case, Oryx has been chosen to create the visual editing tool for the WoAD mechanisms, since JBoss Guvnor[4] provides a visual editing tool that is conceived mainly for the developers.

The first step that will be performed, starting from the document templates editor, is to allow for the creation of the instances of the documents (the 'passive' part of the document), with the retrieving of the associated data, if any. Afterwards, the goal to reach is the full integration with the mechanisms (the 'active' part of the document). Once these works will be accomplished, some validation

[3] http://bpt.hpi.uni-potsdam.de/Oryx/

[4] http://www.jboss.org/drools/drools-guvnor.html

sessions will be started in order to improve the proposed solutions on the basis of the collected results and suggestions.

4 Expected Contributions

Contributions would be expected about how this work could be better situated inside the EUD research field. If similar topics have already been addressed that escaped our attention, a great contribution could be to know which were their principal issues and the solutions that have been conceived to solve them.

Important contributions would regard how to improve the mechanisms editor, in order to lead to a more intuitive, flexible and easy to use rules composer. Until now, only the documents containing a form can be involved into the mechanisms, both in their definition and execution. An important contribution could show how to extend the definition of the mechanisms, allowing the use of different types of document contents (e. g., documents containing diagrams, like a workflow).

Finally, the contributions could be related to improve the usability and effectiveness of the user interface of both the visual editing tools, and consequently the end-user experience.

References

1. Braa, K., Sandahl, T.: Introducing Digital Documents in Work Practices - Challenges and Perspectives. Group Decision and Negotiation 9(3), 189–203 (2000), http://dx.doi.org/10.1023/A:1008783106613
2. Cabitza, F., Gesso, I.: Web of Active Documents: An Architecture for Flexible Electronic Patient Records. In: Fred, A., Filipe, J., Gamboa, H. (eds.) Biomedical Engineering Systems and Technologies. Communications in Computer and Information Science, vol. 127, pp. 44–56. Springer, Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-18472-7_4
3. Cabitza, F., Simone, C.: Active artifacts as bridges between context and community knowledge sources. In: Carroll, J.M. (ed.) C&T, pp. 115–124. ACM, New York (2009)
4. Cabitza, F., Simone, C.: LWOAD: A specification language to enable the end-user develoment of coordinative functionalities. In: Pipek, V., Rosson, M.B., de Ruyter, B., Wulf, V. (eds.) IS-EUD 2009. LNCS, vol. 5435, pp. 146–165. Springer, Heidelberg (2009)
5. Cabitza, F., Simone, C.: WOAD: A framework to enable the end-user development of coordination oriented functionalities. Journal of Organizational and End User Computing 22(1) (2009)
6. Dourish, P., Edwards, W.K., LaMarca, A., Lamping, J., Petersen, K., Salisbury, M., Terry, D.B., Thornton, J.: Extending document management systems with user-specific active properties. ACM Trans. Inf. Syst. 18(2), 140–170 (2000), http://dx.doi.org/http://doi.acm.org/10.1145/348751.348758
7. Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-User Development: An Emerging Paradigm. In: End User Development, vol. 9, ch. 1, pp. 1–8. Kluwer Academic Publishers, Dordrecht (2006), http://dx.doi.org/10.1007/1-4020-5386-X_1
8. Nonaka, I., Takeuchi, H.: The knowledge-creating company: How Japanese companies create the dynamics of innovation. Oxford University Press, USA (1995)

End-User Design in Virtual Worlds: Development of Theory and Virtual Design Environments

Benjamin Koehne

Donald Bren School of Information and Computer Sciences,
University of California, Irvine
6210 Donald Bren Hall
Irvine, CA 92697-3425 USA
bkoehne@ics.uci.edu

Abstract. Understanding end-users not merely as passive consumers of technology but as intelligent actors that can play an active role during the design of interactive technology has changed how technology is adopted and designed. The proposed research focuses on unpacking the transition points that define the steady involvement of novice end-users in technology design. Virtual world environments are chosen to study end-users engaged in design practices. The implementation of a virtual design environment for developing novel, human-centered design approaches is proposed.

Keywords: Meta-design, virtual worlds, human-centered and end-user design.

1 Introduction

The field of end-user design has inspired designers in software engineering and other fields to create information systems and tools that can be modified, extended and even re-invented by its users [4,8]. The vision of open and evolvable systems that provide socio-technical environments to promote end-user participation and collaboration has been expressed in the meta-design framework [3,5].

The ubiquitous extension of our daily lives into virtual spaces provides a challenge and opportunity for researchers of end-user design. Unpredictable use contexts and collaborative processes span around the globe and online. It is time to engage in renewed efforts to revisit established concepts, to extend theories of end-user design, and to create tools and information systems that fit contemporary requirements of professionals and everyday users engaged in highly networked activities.

The increasing popularity and technical sophistication of virtual worlds, such as Second Life[1] and World of Warcraft[2], can be taken as examples for the current shift in human society that plays out on a new socio-technical level. Virtual worlds open up new and exciting interaction spaces and offer unique aspects for scientific research

[1] See: http://www.secondlife.com

[2] See: http://us.blizzard.com/en-us/company/press/pressreleases.html?101007

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 391–394, 2011.

[1]. Virtual worlds allow the observation of user participation processes in design activities on various levels. Massively-multiplayer online role playing games create a carefully crafted fantasy environment with a relatively strict regulatory framework. Open-ended virtual worlds, such as Second Life, allow users a maximum of creativity in designing digital artifacts and forming the environment. Virtual worlds not only attract designers, but mostly everyday casual users with diverse backgrounds and interest. This a unique opportunity for us to study collaborative end-user design processes in these novel and virtually extended and merging contexts of the physical world.

The open-source server platform OpenSimulator[3] in particular affords the replication of environments for end-user design. A sophisticated back-end system allows for a careful analysis of users interacting in the virtual world. With a custom virtual design environment we can inform the development of tools and techniques supporting human-centered design in an increasingly interconnected world.

By extending the study of end-user design processes to virtual worlds we hope to gain new insights into knowledge sharing processes among collaborators [11],highly dynamic collaboration processes [2] and related fields.

2 Preliminary Studies in Two Distinctive Virtual Worlds

We have conducted two preliminary studies in two distinctive virtual worlds to investigate the potential for future research [7]. Our investigations concentrated on the gaming environment Lord of the Rings Online (LOTRO) and the open-ended environment of Second Life. We chose these systems to represent both a gaming-oriented environment and an open-ended virtual world system in our preliminary study. The results serve as a foundation for future research in different virtual environments.

Our investigation in LOTRO and Second Life draws from virtual ethnography [6,10]. Additionally, we draw from multi-sited ethnography [9], which allows us to supplement our findings with interviews with users in the physical world. The ethnographic approach provided us with a qualitative lens to analyze the users' activities in both the gaming-oriented and the open-ended environment.

The goal of the preliminary studies was to illuminate the properties of two distinct virtual worlds that we believe best exemplify concepts of design and the empowerment of end-users. LOTRO and Second Life can individually serve as systems for providing such examples. During our investigation, we focused on the central concepts of meta-design [5]. A synthesis of the exemplified concepts allows for an informed argument for the adoption of virtual world systems to broaden the theory of meta-design.

Further analysis of LOTRO has allowed us to gain insights into mediation aspects of virtual worlds. A socio-technical model of online identity formation and mediation illuminates the processes of identity design in virtual worlds and demonstrates the influences of social and technological structures on self-presentation and interaction. Virtual worlds become extensions of the real world and technical structures provide anchor points for mediation of activities and identity projections.

[3] See: http://www.opensimulator.org

Our investigations into these two virtual worlds have provided us with important insights into the social and technical structures of virtual worlds. We learned that virtual worlds should not be seen as separate places but as natural extensions of the real world. Social networks and scaffolding systems in virtual worlds provide important structural components that allow users to mediate their actions between the real world and the virtual world. Mediation and identity formation is consciusly performed by many of the users that we observed and interviewed.

3 Proposed Research: Continued Qualitative Investigations and Virtual Design Environment

Based on our preliminary studies, the main research question is formulated as such: How can information systems and interactive technologies support *the empowerment of (novice) end-users* and how can software engineers and designers of interactive technologies leverage and *employ social and technical structures* that enable users to dynamically switch roles to become *voluntary designers*?

The research challenge represents the provision of frameworks and design process models that provide designers with the necessary means to create systems and technologies that incorporate guidance and interpretive flexibility at the same time. Virtual worlds represent an opportunity to extend existing design theories that already focus on how designers and users interact in highly dynamic design/use contexts.

The new perspectives gained from the preliminary studies allow us to expand our research approach with the clear directive of the research question as the underlying goal. Continued qualitative investigations in public virtual worlds and interviews with designers in Second Life provide the basis for the extension of end-user design theory.

A virtual design environment is implemented in OpenSimulator allowing for user studies to verify and extend our findings from the qualitative investigations in public virtual worlds. OpenSimulator represents an open-source server platform for virtual worlds. The system is sufficiently flexible to allow for the implementation of various design settings and the provision of virtual design products to potential users.

The virtual design environment in OpenSimulator can be configured to represent the special needs of certain focus groups such as software engineers and programmers. Data is gathered by directly logging system outputs of the server system to a database or by recording interaction in the virtual world on video. Follow-up interviews will provide additional qualitative data to contextualize the findings.

4 Contributions

The proposed research makes several important contributions. The investigations develop a better theoretical understanding of novice users (i.e. users initially unaware of their development opportunities) that become empowered to take part in design processes. The complex properties of the social context and the embodiment of end-users in design environments are unpacked as afforded by the unique virtual fieldsites.

The findings contribute to the development of tools and design environments in virtual worlds that facilitate the dynamic processes between design- and use-time.

With the previously defined main research question in mind, the proposed research specifically aims to have a positive impact on virtual educational environments and technology-supported collaborative work environments.

Acknowledgements

This work was supported by the U.S. National Science Foundation under grant #0808783. I would like to thank David Redmiles, Gerhard Fischer and Matthew Bietz and the CRADL research group at UC Irvine for their very helpful feedback.

References

1. Bainbridge, W.S.: The Scientific Research Potential of Virtual Worlds. Science 317(5837), 472–476 (2007)
2. Churchill, E.F., Snowdon, D.N., Munro, A.J.: Collaborative virtual environments: digital places and spaces for interaction. Springer, London (2001)
3. Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A.: A meta-design approach to end-user development. Paper presented at the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (2005)
4. Costabile, M.F., Mussio, P., Provenza, L.P., Piccinno, A.: End users as unwitting software developers. In: Proceedings of the 4th International Workshop on End-User Software Engineering, ACM, Leipzig (2008)
5. Fischer, G., Scharff, E.: Meta-design: design for designers. In: Proceedings of the 3rd Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques. ACM, New York City (2000)
6. Hine, C.: Virtual ethnography. Sage, London (2000)
7. Koehne, B., Redmiles, D., Fischer, G.: Extending the Meta-Design Theory: Engaging Participants as Active Contributors in Virtual Worlds. In: Costabile, M.F., et al. (eds.) IS-EUD 2011. LNCS, vol. 6654, pp. 264–269. Springer, Heidelberg (2011)
8. Lieberman, H., Paterno, F., Klann, M., Wulf, V.: End-user development: An emerging paradigm. Kluwer Publishers, Dordrecht (2006)
9. Marcus, G.E.: Ethnography in/of the World System: The Emergence of Multi-Sited Ethnography. Annual Review of Anthropology 24, 95–117 (1995)
10. Nardi, B.A.: My life as a night elf priest: an anthropological account of world of warcraft. The University of Michigan Press, Ann Arbor (2010)
11. Teasley, S., Covi, L., Krishnan, M.S., Olson, J.S.: How does radical collocation help a team succeed? In: Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work. ACM, Philadelphia (2000)

Development Tools for Interactive Behaviors

Stephen Oney

Carnegie Mellon University
Human-Computer Interaction Institute
5000 Forbes Ave. Pittsburgh, PA, USA 15232
soney@cs.cmu.edu

Abstract. This research uses participatory design workshops and user-centered design with trained interaction designers to guide the development of a new programming language and environment for creating interactive applications. Interactive behaviors, which define the operation of an interactive application, are usually difficult for interaction designers to program because many interaction designers do not have formal programming training and many features of interactive behaviors make the task of programming them distinct from, and often more challenging than, other programming tasks. This research aims to create a programming language and environment that is tailored to the needs of interaction designers and that alleviates the problems that make programming interactive behaviors difficult.

Keywords: end-user programming, interaction design.

1 Introduction

Rogers, Sharp, & Preece define interaction design as “designing interactive products to support the way people communicate and interact in their every day and working lives.” [1] Interaction designers are often tasked with designing novel and complex interactive software as part of their job. The medium of software presents a unique challenge for interaction designers who are interested in writing interactive applications. Unlike other designers, who work with their materials in a studio or workshop, interaction designers are not able to engage in meaningful reflection-in-action [2] (which means to evaluate and generate ideas while in the process of creating) when designing interactive software. In addition, the threshold of programming knowledge required to programmatically create new interactive behaviors is prohibitively high for many interaction designers, which often forces interaction designers to rely on professional developers to program the interactive behaviors they design.

In my research, I am interested in investigating ways to enable and encourage interaction designers themselves to design and *develop* interactive software. This means accounting for not only the needs of interaction designers, but also analyzing the features of interactive behaviors that make them difficult to program in traditional programming languages and exploring ways to lower the threshold for creating interactive software. This is a form of End-User Development since interaction designers are authoring code, but they are not professional programmers.

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 395–398, 2011.

2 Background

A recent survey of interaction designers shows that interaction designers find it more difficult to prototype and implement the *feel* of an interactive application than the *look* [3]. Further, 78% of the participants in this survey indicated that designing interactive behaviors requires collaborating with a developer. Designers often communicate a design to a developer through annotated design sketches and storyboards, but they indicated that communication breakdowns are frequent [3].

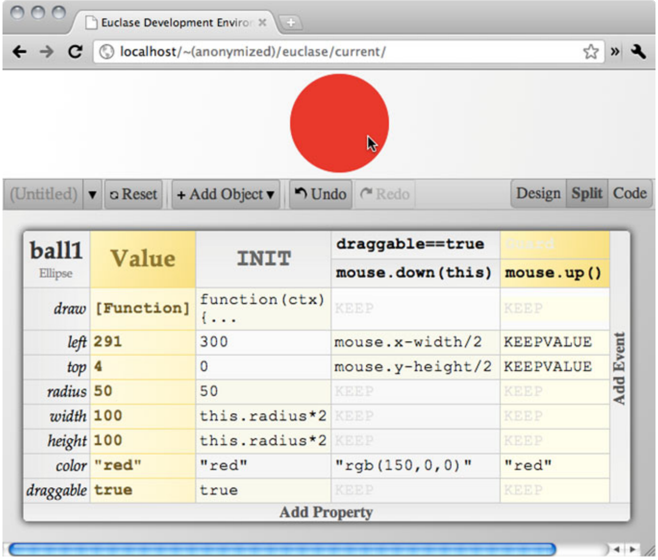
Even putting aside the possibility of communication breakdowns and the cost of having to collaborate with professional developers, relying on another team member to prototype and implement their interactive behaviors reduces their potential for reflection-in-action and to iterate on and evaluate their design. So why do not more interaction designers learn to program? In a different survey, they pointed to the high learning curve, time consumption, the difficulty of creating novel interfaces, problems with generated code, and toolset limitations as weaknesses of various programming languages and environments [4]. Additionally, from a software engineering perspective [5], the task of writing interactive applications presents a unique set of challenges, as I will outline in the next section.

3 Research Approach

This research includes participatory design workshops conducted with interaction designers to gain insight into design requirements, the design of the language & environment, and evaluation & iteration through evaluative user studies of environment prototypes.

In the participatory design workshops, which were conducted with fourteen interaction designers and programmers with at least two years of professional experience, and described in detail in [6], designers indicated the need to better evaluate their designs, and the importance of examples for exploration and communication and of programming tools that can keep track of design rationale.

In designing the language and environment, I focused on five features of interactive behaviors that make creating interactive applications difficult. First, interactive behaviors are usually graphical in nature, and while it is relatively easy to declaratively specify the look of an application, imperatively writing a graphical application that controls how it operates is difficult. Second, interactive behaviors are often state-oriented; their behavior may be dependent on a combination of global and local states. Third, interactive behaviors are often constraint-heavy; conceptually, there are often constraints that update a view based on some underlying model, and there are constraints on the layout of elements in the view with respect to each other. Fourth, interactive behaviors are often event-based, as they react to user input. Finally, interactive behaviors are often integrated with animations, and coordinating the behavior with the animation is often a significant challenge.



ball1 Ellipse	Value	INIT	draggable==true	Guard
			mouse.down(this)	mouse.up()
draw	[Function]	function(ctx) {...	KEEP	KEEP
left	291	300	mouse.x-width/2	KEEPVALUE
top	4	0	mouse.y-height/2	KEEPVALUE
radius	50	50	KEEP	KEEP
width	100	this.radius*2	KEEP	KEEP
height	100	this.radius*2	KEEP	KEEP
color	"red"	"red"	"rgb(150,0,0)"	"red"
draggable	true	true	KEEP	KEEP

Fig. 1. A representation of a draggable red circle. The top half of the window shows the "design" view, while the bottom half shows the "code" view. In the code view, attribute names are shown in the far left column. The current values of the attributes are shown in "Value" column. Initial values are shown in the `INIT` column. The subsequent two columns specify constraints that will hold in various states: the fourth column specifies constraints that will hold when the user is dragging the circle (to constrain the center of the circle to always be the mouse location), and the last column specifies constraints that will hold after the user stops dragging (`KEEPVALUE` keeps the current value but gets rid of the constraints that were in place when the user was dragging the circle).

4 Progress

The current iteration of a prototype of our language and environment is shown in Fig. 1 above. Interactive behaviors are written declaratively; every object has a set of attributes that can be static values (`3`, `red`, etc.) or constraints (`this.x+foo.bar`, `max(a,b)`, etc.). Attributes of objects are represented by rows in the object. Events are represented as columns, with the values in a column specifying the constraints that will hold after that event occurs. Our prototype is implemented in client-side JavaScript and provides immediate feedback as the user edits the code. An initial user test conducted with interaction designers showed promise; interaction designers took advantage of the immediate feedback that our prototype provides when they updated their code. They also were very willing to experiment in their code. In fact, some designers asked for more immediate feedback where the design view would update as they were typing code.

For future work, I plan on making additions to the prototype, including a timeline view for specifying and coordinating animations, a state-flow diagram view to illustrate states and transitions between states, and improving the usability of the language syntax through iterative usability evaluation. Another addition that I plan to make for the environment is to create an “open box” widget set. One of the strengths interaction designers see in tools like Adobe Flash Catalyst is the availability of widgets to help them get started [4]. However, widgets in such tools are usually inflexible, and cannot be customized. An open box widget set would include pre-provided widgets that encourage designers to extend and customize them.

5 Impact for End-User Development

This research will result in the creation of, and design recommendations for, programming languages and environments for creating interactive behaviors. The two-dimensional representation that the current prototype uses is a unique contribution that may prove to be a simpler representation for interactive behaviors than the style of imperative code used by C-derived languages like Processing and OpenFrameworks[1]. Although previous research has focused on providing widgets, or programming-by-example tools to reduce the threshold of creating interactive applications, my research focuses on the underlying representation of interactive behaviors. While interaction designers have played a large part in the design of this environment, its usefulness will likely extend beyond interaction designers. I plan on releasing the development environment for general use over the web.

Acknowledgments. I thank my advisor, Brad Myers, for his continuing guidance and support. The author is also supported by the Ford and ARCS foundations. This research has been partially funded by NSF grants IIS-0757511 and CCF-0811610.

References

1. Rogers, Y., Sharp, H., Preece, J.: Interaction Design: Beyond Human-Computer Interaction. John Wiley & Sons, Ltd., West Sussex (2007)
2. Schön, D.: The Reflective Practitioner: How professionals think in action (1983)
3. Myers, B.A., Park, S.Y., Nakano, Y., Mueller, G., Ko, A.J.: How Designers Design and Program Interactive Behaviors. In: VL/HCC, pp. 177–184 (2008)
4. Carter, A., Hundhausen, C.: How is User Interface Prototyping Really Done in Practice? A Survey of User Interface Designers. In: VL/HCC, pp. 207–211 (2010)
5. Letondal, C., Chatty, S., Phillips, G., Fabien, A., Conversy, S.: Usability requirements for interaction-oriented development tools. Psychology of Programming (2010)
6. Ozenc, F.K., Kim, M., Zimmerman, J., Oney, S., Myers, B.: How to support designers in getting hold of the immaterial material of software. In: CHI, pp. 2513–2522 (2010)

[1] http://processing.org/ & http://www.openframeworks.cc/

Creating Useful, Usable and Accessible VR Design Tools: An EUD-Based Approach

J.P. Thalen and M.C. van der Voort

Laboratory of Design Production and Management
University of Twente, 7500 AE Enschede, The Netherlands
j.p.thalen@utwente.nl, m.c.vandervoort@utwente.nl

Abstract. Virtual Reality (VR) tools create an alternative reality in which worlds, objects and characters can be experienced that may not yet be experienced in reality. As such, VR can help product designers in the early stages of the product development process with evaluating virtual product concepts. The current set of VR tools and VR development toolkits however targets programmers and computer experts rather than product designers, thus limiting the adoption of this technology in the field of product design. The research presented in this paper applies End-User Development (EUD) principles to let designers describe, create and evaluate VR *design tools* that are useful, usable and accessible.

Keywords: virtual reality, product design, tool development.

1 Introduction

The product development process can be described as a sequence of gathering requirements, generating and selecting concepts, engineering the product and finally releasing the product to the market. One of the challenges in the early stage of product development is the lack of concrete design information; ideas are discussed on a conceptual level. Even well informed fellow designers or engineers can make false assumptions or misinterpret a concept. This problem grows when less informed stakeholders such as suppliers, clients or end-users, are involved.

We propose to use Virtual Reality (VR) as a common language in early stages of product development to facilitate communication between designers and stakeholders. VR technologies create an alternative reality in which worlds, objects and characters can be experienced that may not yet be experienced in reality. A virtual representation of a future *product* or a future *use context* can help the dialog between designers and stakeholders. A drive simulator for instance provides a virtual context in which new automotive concepts can be evaluated without putting a test participant in danger[6], and without creating expensive physical prototypes [4].

While this example illustrates advantages of VR *applications* in the early stages of the product development process, the actual use of VR is still quite limited. Making VR accessible for product designers requires appropriate *VR development tools*. This paper argues that the current set of VR development

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 399–402, 2011.

tools (see [7] for an extensive overview) does not match the skills and requirements of product designers. Consequently we propose a development approach in which EUD principles are used to create more *useful, usable* and *accessible* VR development tools.

2 VR Development Tools

The current state of adoption of VR in the product development process was investigated through a series of in-depth interviews with engineers, managers and designers from four different companies (automotive, mechatronics, electronics and mechanical systems design)[1]. This study showed that the threshold for using VR in the product development process depends on the usability and accessibility of VR development tools. Development tools often originate from research in computer science (for instance [1], [2] and [5]) and provide a good platform for further development by experts, but they are by no means usable by non-expert end-users (e.g. product designers with no or limited programming skills).

Involving the end-users (e.g. product designers) in the development of VR tools ensures that the tools fulfill a useful purpose and fit the skills of the end-user. Continuous end-user involvement also creates awareness of tool opportunities, restrictions and latent requirements. Capturing latent requirements is especially important given the end-user's unfamiliarity with VR. Section 3 explains how EUD principles are applied to the development approach for VR design tools.

3 Approach

The participation of three companies[2] allows us to directly involve end-users (e.g. product designers from these companies). The end-users will be involved in matching company requirements to available VR technologies and in defining the user interface for the tool. As such, EUD is used *during the design phase*[3] as opposed to *during usage*. The approach consists of two major phases.

In the first phase three task specific VR development tools will be developed for the three different companies involved. These tools will support design tasks and address challenges identified within the company. End-users control this development phase by matching their interests to specific VR technologies (see section 4). For instance, company A could be interested in using augmented reality in concept evaluation, or company B is interested in using a motion capture tool in an ergonomics analysis.

In the second phase we aim to find out how well the company-specific VR tools translate to the product development process of other companies. EUD is applied here to let end-users explore and evaluate the VR design tools from their own point of view. For instance, an augmented reality tool developed for

[1] Conducted in the Netherlands, between November 2009 and March 2010.

[2] These are three of the four companies also involved in the interviews.

company A could also be useful for company B if tool parameters X and Y are modified appropriately. The results of phase two will show whether or not it is worthwhile to pursue a 'one-for-all' VR development tool or rather go for a suite of various VR development tools.

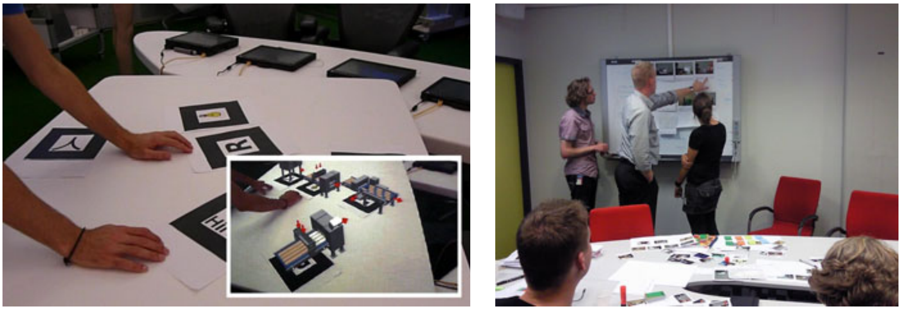
4 Preliminary Results

The approach presented in section 3 is currently being carried out with three companies. This section outlines the first experiences with involving end-users in the development of the company specific VR tools.

Prior to phase 1, a *VR demo session* was organized to create a common understanding of VR and VR design tools. This session involved representatives from all three companies. End-users (product designers) were shown various forms of VR technologies being used in design applications, including an augmented reality application (see figure 1(a)), an immersive drive simulator, a 3D virtual usability test lab and a 3D interactive 'experience' lab. The end-users were involved in determining the functionality of the demos through interviews as well as less formal meetings. All demo applications allowed end-users to try them out themselves during the demo session. Throughout the workshop end-users were encouraged to discuss not only their own demonstrations, but also watch the other demonstrations.

Having established a common understanding of VR design tools within the group of companies, research focused on the first company to initiate phase 1. A workshop was organized to gather requirements and specifications for a VR development tool, without writing a single line of code. End-users created visual storyboards to describe the functional behavior of the VR development tool they have in mind (see figure 1(b)).

For the first company this workshop led to the definition of an augmented reality tool that lets consumers evaluate future products in a realistic environment.



(a) One of the VR demos shown during the demo session.

(b) Three end-users discuss and present their visual storyboard.

Fig. 1. Photos from the VR demo session and the group workshop

Future efforts will elaborate on this idea by creating a functional prototype that can be evaluated by the company as well as the other industrial partners.

5 Expected Contributions and Future Work

The approach presented in this paper involves end-users throughout the definition, development and evaluation of VR design tools. Initial results show that after establishing a common understanding of VR design tools, end-users were able to define clear tool expectations, requirements and specifications in a group workshop. Short term future work will focus on creating a functional tool prototype for the first company, based on the results of the workshop. After validating this prototype within the company, in the long term we also aim to identify the parameters that make the tool sufficiently flexible to be deployed in other companies. To explore these possibilities we intend to make further use of EUD principles by letting product designers deploy and evaluate the VR tools in a new company context.

Acknowledgements. The authors gratefully acknowledge the support of the Innovation-Oriented Research Programme Integral Product Creation and Realization (IOP IPCR) of the Netherlands Ministry of Economic Affairs, Agriculture and Innovation.

References

1. Bierbaum, A., Just, C., Hartling, P., Meinert, K., Baker, A., Cruz-Neira, C., juggler, V.R.: a virtual platform for virtual reality application development. In: ACM SIGGRAPH ASIA 2008 Courses, Singapore, pp. 1–8 (2008)
2. Carlsson, C., Hagsand, O.: DIVEA platform for multi-user virtual environments. Computers & Graphics 17(6), 663–669 (1993)
3. Lieberman, H., Patern, F., Wulf, V.: End User Development, vol. 9. Springer, Netherlands (2006)
4. Miedema, J., van der Voort, M.C., van Houten, F.J.A.M.: Advantageous application of synthetic environments in product design. CIRP Journal of Manufacturing Science and Technology 1(3), 159–164 (2009)
5. Reitmayr, G., Schmalstieg, D.: An open software architecture for virtual reality interaction, Baniff, Alberta, Canada, pp. 47–54 (2001)
6. Tideman, M., van der Voort, M.C., van Houten, F.J.A.M.: A new product design method based on virtual reality, gaming and scenarios. International Journal on Interactive Design and Manufacturing 2(4), 195–205 (2008)
7. Wright, T.E., Madey, G.: A survey of technologies for building collaborative virtual environments (2009)

A Meta-design Framework to Support Multidisciplinary Teams' Online Collaboration

Li Zhu

Dipartimento di Informatica e Comunicazione, Università degli Studi di Milano,
Via Comelico 39/41, 20139 Milano, Italy
zhu@dico.unimi.it

Abstract. The ever-growing complexity of design projects needs the collaboration of multidisciplinary design teams. Communication gaps arise between stakeholders who belong to different design communities. Moreover, the co-evolution of design communities and their systems requires an open environment to support emerging needs. The Hive-Mind Space (HMS) model has been proposed to support cultures of participation and to tackle the co-evolution of users and systems. MikiWiki is an HMS prototype, implemented to explore the use of boundary objects and habitable environments to facilitate communication and tackle co-evolution of users and system.

Keywords: HMS model, Meta-design, End User Development, Boundary Objects, Habitable Environment, Co-evolution.

1 Research Problem

The ever-growing complexity of design projects needs the collaboration of multidisciplinary design teams. Web 2.0, social media and advanced information technology enable and foster this cross-culture and cross-domain collaboration.

However, communication often breaks down due to the fact that stakeholders from different cultures and backgrounds utilize different notations and might have different interpretations of the same thing. Moreover, the co-evolution of users and systems [1] as well as the co-evolution of problems space and solutions space [2] challenge traditional collaboration systems.

My research questions are: how can we provide a socio-technical collaboration environment to bring multidisciplinary design communities together and support their diverse and evolving design activities? How can we support design communities' communication via digital artifacts in a flexible manner?

2 Theoretical Background

This is an interdisciplinary research project stemming from several related concepts. It addresses End-User Development (EUD) [3], meta-design [4], and social creativity, and it focuses on providing online tools to support diverse design communities.

The emerging paradigm of EUD aims to explore opportunities to enable software development at use time. EUD techniques propose various approaches that allow

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 403–406, 2011.

users of software systems, who are not professional software developers, to create, modify or extend software artifacts at use time [3]. However, EUD tends to solve the co-evolution [1] problem from a technical perspective and mainly focuses on tailorability rather than other related social issues [5].

The meta-design approach, on the other hand, aims to tackle emerging behavior by providing socio-technical environments to empower users to be active knowledge contributors [4]. The SER (Seeding, Evolutionary growth, Reseeding) three-phase process model [6] is a meta-design model that suggests that systems that evolve over a sustained time span must continually alternate between periods of unplanned evolution and periods of deliberate restructuring and enhancement. The Software Shaping Workshop (SSW) is another meta-design model [7] where the system has three design levels and is organized into "workshops" dedicated to different communities. Design, implementation and tailoring of workshops are incremental, since communities design perpetual-beta artifacts and not final products [8].

In addition, the dynamic and complex nature of collaborative design problems requires flexible and innovative responses. Social creativity is not a luxury but a necessity to cope with emerging problems in the course of collaborative activities [6]. The knowledge associated with the design problems however is tacitly distributed among the various Communities of Practice (CoP) [9]. Bringing divergent viewpoints together and creating a shared common understanding will help CoPs to gain new insights, and hence to solve problems more creatively [10].

3 Research Approach

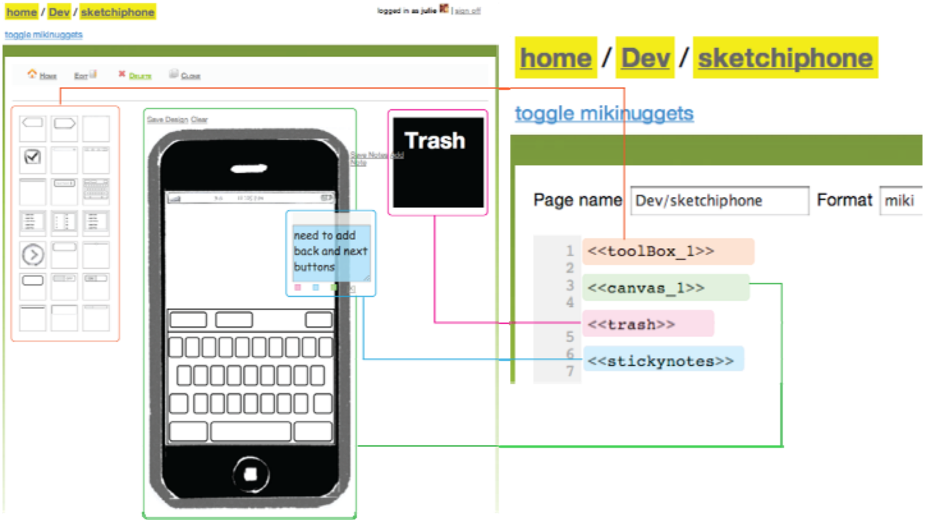
My research makes use of mixed methodology, combining two approaches: one rooted in computer science to provide a system solution dealing with co-evolution, for future computational artifacts, and the other approach rooted in social science tradition to provide a deeper understanding of existing work practices and processes by taking an ethnographic approach [11].

Observations of offline collaboration help us to better understand communication, negotiation processes and emergent creativity during collaboration: simple and small design tools, such as colored pencils, paper and post-it notes can support very rich and complex collaboration and communication. These collaborative patterns and dynamic behaviors can be understood and introduced into the software system.

4 Current Status of My Research

Based on the SSW, the Hive-mind Space Model (HMS) has been proposed to support multidisciplinary design teams' collaboration as well as to foster their creativity [12]. The HMS model inherits the SSW features and supports communication by introducing the concepts of boundary object [13], boundary zone [16] and localized habitable environments [15]. Boundary objects are used to mediate communication. Environments provide CoPs with specific affordances that support communities' collaborative practice and provide them with suitable languages and tools to foster their personal and common needs.

Each CoP is locally controlled but globally interconnected to each other, forming a Community of Interest (CoI) [10]. The HMS model encourages collective knowledge aggregation and social creativity by providing CoPs with the opportunity to shape their environment.


home / Dev / sketchiphone
toggle mikinuggets
logged in as julie | sign off
Home
Edit
Delete
Close
Save Design Clear
Save Notes add Note
Trash
need to add back and next buttons
home / Dev / sketchiphone
toggle mikinuggets

Page name Dev/sketchiphone Format miki

```
<<toolBox_1>>

<<canvas_1>>

<<trash>>

<<stickynotes>>

```

Fig. 1. MikiWiki iPhone mockup page

We developed MikiWiki, a Ruby web application, to prototype the main features of the HMS model. Mikinuggets are the building blocks of MikiWiki and they act as boundary objects within and between CoPs. Users can design or tweak existing boundary objects to solve emergent problems during collaboration. Fig. 1 shows an iPhone wireframe page that simply includes four mikinuggets, namely toolbox, canvas, trash and stickynotes.

MikiWiki provides a common collaboration context across the system and opportunities for design communities to build domain-oriented environments where they can work while being aware of the activities of others. Instead of merely providing tools for text content production as traditional wikis, MikiWiki allows different CoPs to collaborate in practice design and to continuously evolve the whole wiki system.

The next step will focus on providing a UI mockup design environment in order to bring software engineers, designers and end users together and to support their different levels of participation in design. The other activity will be a story-game where users can collaboratively create a shared story. The evaluation process will focus on how mikinuggets as shared boundary objects and habitable environments can be used to support and shape collaboration and communication.

My plans are to finalize the MikiWiki prototype implementation by February 2011, complete the feedback data analysis and prototype modification by June 2011, finish the dissertation in December 2011, and defend my dissertation by April 2012. The contribution of this work is to improve the SSW model and to provide means of communication within the meta-design model to enhance design communities' creativity as a whole.

Acknowledgments. The work of Li Zhu was supported by the Initial Training Network "Marie Curie Actions", funded by the FP 7 - People Programme with reference PITN-GA-2008- 215446 entitled "DESIRE: Creative Design for Innovation in Science and Technology."

References

1. Bourguin, G., Derycke, A., Tarby, J.C.: Beyond the Interface: Co-evolution inside Interactive Systems - A Proposal Founded on Activity Theory. In: Blandford, A., Vanderdonckt, J., Gray, P. (eds.) Proc. of IHM-HCI 2001, pp. 297–310. Cépaduès-Éditions, Toulouse (2001)
2. Dorst, K., Cross, N.: Creativity in the design process: co-evolution of problem–solution. Design Studies 22(5), 425–437 (2001)
3. Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-User Development: An Emerging Paradigm. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development, pp. 1–8. Springer, Dordrecht (2006)
4. Fischer, G.: Social Creativity, Symmetry of Ignorance and Meta-design. Knowledge-Based Systems Journal 13(7-8), 527–537 (2000)
5. Pipek, V.: From tailoring to appropriation support: Negotiating groupware usage. PhD Thesis, University of Oulu, Oulu (2005)
6. Fischer, G.: Seeding, Evolutionary Growth and Reseeding: Constructing, Capturing and Evolving Knowledge in Domain-Oriented Design Environments. Automated Software Engineering 5, 4 (1998)
7. Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A.: Visual Interactive Systems for End-User Development: a Model-based Design Methodology. IEEE TSMCA 37(6), 1029–1046 (2007)
8. Bruns, A.: Blogs, Wikipedia, Second Life, and Beyond: from production to produsage. Peter Lang, New York (2008)
9. Wenger, E.: Communities of Practice. Learning, Meaning, and Identity. Cambridge University Press, Cambridge (1998)
10. Fischer, G.: Communities of Interest: Learning through the Interaction of Multiple Knowledge Systems. In: Proc. of IRIS 2001, pp. 1–14 (2001)
11. Schmidt, K.: Riding a tiger, or computer supported cooperative work. In: ECSCW 1991, Proc. of the Second European Conference on Computer Supported Cooperative Work, vol. 5(2-3), pp. 1–16 (1991)
12. Zhu, L., Mussio, P., Barricelli, B.R., Iacob, C.: A Habitable Space for Supporting Creative Collaboration. In: Proc. of CTS 2010, pp. 617–622 (2010)
13. Star, S.L., Griesemer, J.R.: Translations and Boundary Objects: Amateurs and Professionals in Berkley"s Museum of Vertebrate Zoology, 1907-1939. Social Studies of Science 19(3), 387–420 (1989)
14. Mørch, A., Andersen, R.: Mutual development: A case study in customer-initiated software product development. In: Pipek, V., Rosson, M.B., de Ruyter, B., Wulf, V. (eds.) IS-EUD 2009. LNCS, vol. 5435, pp. 31–49. Springer, Heidelberg (2009)
15. Barricelli, B.R., Marcante, A., Mussio, P., Parasiliti Provenza, L., Valtolina, S., Fresta, G.: BANCO: a Web architecture supporting unwitting end-user development. IxD&A - Interaction Design & Architecture(s): Design for the Future Experience 5-6, 23–30 (2009)

Part V
Workshops

European-American Collaboration Workshop

Christopher Scaffidi[1], Margaret Burnett[1], Maria Francesca Costabile[2],
Simone Stumpf[3], and Volker Wulf[4]

[1] Oregon State University, School of Electrical Engineering
and Computer Science, Corvallis, OR, 97331-4501, USA
[2] Università degli Studi di Bari,Dipartimento di Informatica, 70125 Bari, Italy
[3] City University London, Centre for HCI Design, London, EC1V 0HB, UK
[4] University of Siegen, Media Research Institute, 57068 Siegen, Germany
{cscaffid,burnett}@eecs.oregonstate.edu,
costabile@di.uniba.it, simone.stumpf.1@city.ac.uk,
volker.wulf@uni-siegen.de

Abstract. Different researchers have focused on complementary aspects of end-user development. For example, some people work on tool-building while others explore different models for participatory programming. This workshop will focus discussion on identifying opportunities for collaboration, particularly between European and American research groups. Submitted position papers have revealed three topics that could be used to frame collaborative projects. At the workshop, participants will meet one another, discuss how to pursue collaboration, and explore avenues for obtaining funding to support international collaboration.

Keywords: Design, Human Factors, Languages, End-user development, Collaboration, Programming Environments, Social Technologies.

1 Introduction

Researchers in both Europe and America have invested over a decade of work in developing approaches to support end users who actively participate in the development of software, so that the final products can be more suitable to their needs and expectations. Many of these lines of inquiry are complementary to one another. For example, some research is focused on settings where endusers, acting as non-professional programmers, work independently or perhaps in concert with one another, but not with any direct assistance from professional programmers. Other research focuses on contexts where endusers work in partnership with professional programmers.

These differences in focus are complementary, with numerous as-yet untapped opportunities exist for collaboration among researchers who take these and other complementary approaches. Achieving this collaboration will require some intentional focus and discussion, because the researchers performing work in this area are scattered over both Europe and America. In order to lay a foundation for such collaboration, particularly between European and American groups, IS-EUD 2011 will host a

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 409–412, 2011.

workshop specifically focused on identifying opportunities for collaboration. The workshop will be held June 7 in Torre Canne (Brindisi), Italy.

2 Audience and Topical Focus

Attendance is anticipated from researchers interested working in human-computer interaction, software engineering, computer-supported collaborative work, and related areas.

Five participants submitted position papers (which was optional). The full position papers are available online at the workshop website:

< http://web.engr.orst.edu/~cscaffid/gather/iseud2011/ >

These papers signaled interest in collaborating on projects related to three particular topics:

- Supporting the efforts of end-user and professional developers to learn knowledge and skills, such as during participatory programming
- Providing more effective tools for end-user development domains, such as mashup-programming and integration of knowledge bases
- Design of social technologies to enhance communication, awareness and understanding, particularly in the context of culturally diverse communities

3 Workshop Plan

The workshop will begin with a 30-minute overview of the workshop objectives and short introductions. For the remainder of the morning, participants will use a systematic “speed dating” process to introduce themselves one-on-one to each other person in the other group, present topics of interest for collaboration, and propose possible ways for carrying out these collaborations

After a lunch break, participants will spend the afternoon in discussion sessions organized around the topics presented in the morning. The focus of these discussions will be to identify opportunities for collaboration in several categories of research, as well as to explore avenues for obtaining funding to support international collaboration. The workshop will conclude by having each break-out group present a summary of opportunities they have identified for integrating research or for collaborating. These summaries will be integrated into a final report.

4 Organizer Backgrounds

Five researchers have served as organizers and members of the program committee.

Christopher Scaffidi (lead organizer) is an Assistant Professor of Computer Science at Oregon State University and is Director of the EUSES Consortium <http://eusesconsortium.org/>, an association of ten institutions. His research focuses on the intersection of human-computer interaction and software engineering. Most of his current projects aim to help software users to create code for themselves and to effectively share code with one another. His committee service includes IS-EUD and

EICS, as well as workshops or meetings at CHI, SPLASH (OOPSLA), and VL/HCC. He earned a Ph.D. in Software Engineering from Carnegie Mellon University and has seven years of professional software engineering experience.

Margaret Burnett, Professor of Computer Science at Oregon State University, was founding Project Director of EUSES from 2003-2009. Her current research focuses on end-user programming, end-user software engineering, information foraging theory as applied to programming, and gender issues in those contexts. She is also the principal architect of the Forms/3 and the FAR visual programming languages and of the WYSIWYT testing methodology for end-user programmers. She was a long-time member of the Steering Committees for IEEE VL/HCC and ACM SoftVis, and has also keynoted, co-chaired and served on organization committees for numerous conferences, including IS-EUD.

Maria Francesca Costabile, Professor of Computer Science at University of Bari, Italy, was responsible of one of the four managing nodes of the EU sponsored network of excellence on End-User Development EUD-net (2002-2004). She has received research grants from many national and international organizations. Interaction design, end-user development, meta-design approaches, user experience are among her current research interests, from both theoretical and applicative points of view. Prof. Costabile is regularly in program committees of international conferences and workshops. She has been for many years in the Steering Committee of the Advanced Visual Interfaces Conference (AVI), and is in the Steering Committee of Visual Languages and Human Centric Computing Symposium. She is Co-Chair of the Third International Symposium on End-User Development. She has been Program Co-Chair of CHI 2008, Program Co-Chair of Interact 2005 and Program Chair of AVI 2004. She is a founding member of the Italian Chapter of ACM SIGCHI, and served as Chair from 1996 to 2000.

Simone Stumpf, Lecturer in the Centre for HCI Design at City University London, is a member of EUSES. Her current research focuses on end-user programming in the context of intelligent systems, information management, and privacy management. She is also interested in gender issues, particularly concerning privacy. Simone Stumpf received a PhD in Computer Science from University College London. She was previously the workshop organizer for TAKMA, run in conjunction with DEXA.

Volker Wulf is a professor in Information Systems and the director of the Media Research Institute at the University of Siegen. At Fraunhofer FIT, he heads the research group User-Centred Software-Engineering (USE). He is also a founding member of the International Institute for Socio-Informatics (IISI), Bonn. After studying computer science and business administration at the RWTH Aachen and the University of Paris VI., he got a Ph.D. at the University of Dortmund and a habilitation degree at the University of Hamburg, Germany. In 2001, he worked as a research fellow at the Massachusetts Institute of Technology (MIT), Cambridge, MA. In 2006/07 Wulf spent a sabbatical as a Fulbright Scholar at the University of Michigan, Ann Arbor, and at Stanford University, Palo Alto. His research interests lie primarily in the area of Computer Supported Cooperative Work, Knowledge Management, Computer Supported Cooperative Learning, End-User Development, Human Computer Interaction, Participatory Design, and Organizational Computing. He published more than 200 papers. He edited 10 books among which "Expertise Sharing: Beyond Knowledge Management" and "Social Capital and Information Technology" both with MIT Press

Cambridge MA and "End User Development" with Springer Dordrecht are probably best known. As a conference co-chair he hosted the 1st and 2nd International Symposium on End User Development (IS-EUD) in Sankt Augistin (2003) and in Siegen (2009), 11th International Conference on Human-Computer Interaction with Mobile Devices and Service (Mobile HCI 2009) as well as the 7th European Conference on Computer Supported Cooperative Work (ECSCW 2001) in Bonn and Communities & Technologies (C&T 2003) in Amsterdam.

Acknowledgments. We wish to thank the following for their contribution to the success of this conference: European Office of Aerospace Research and Development, Air Force Office of Scientific Research, United States Air Force Research Laboratory <http://www.london.af.mil>

Empowering End-Users to Develop Service-Based Applications

Nikolay Mehandjiev[1], Antonella De Angeli[1,2] , Usman Wajid[1],
Abdallah Namoun[1], and Alberto Battocchi[2]

[1] The University of Manchester, UK
{firstname.lastname}@mbs.ac.uk
[2] University of Trento, Italy
{deangeli,battocchi}@disi.unitn.it

Abstract. The 2nd International Workshop on End User Development for Services (EUD4Services) focuses on the issues encountered when people who are not educated as software developers attempt to create and compose software services, and on approaches and theories aiming to support such activities. The aim is to establish a community of academics and practitioners and facilitate the production of a coherent body of work related to this area.

Keywords: End User Development, service oriented architecture.

1 Introduction

The establishment of the Service Oriented Architecture paradigm in professional software development is opening new challenging environments for the EUD community. Service Oriented Architecture has created and disseminated a large number of reusable software components which can be linked and organised into new and evolving applications to satisfy specific business and personal needs. If, on the one hand, SOA provides promising tools for the EUD agenda, so far the SOA approach has been characterised by a very technical attitude with little, if any, interest to the final user of the resulting applications. Services have been designed to perform software functionalities which can be linked to each other to perform complex tasks, yet the responsibility for composition and deployment was left to expert programmers, who are also assumed to be in charge of designing the interface between services and their users. In this respect, the uptake of EUD within the SOA paradigm is hampered by a number of emerging issues, including intrinsic difficulties stemming from the complexity of technology and distributed nature of computations.

The aim of EUD4Services workshop is to establish a community of academics and practitioners and facilitate the production of a coherent body of work related to this area. Specific lines of research include:

(a) Studies of organisational and societal practices involving the development of service-based software systems;

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 413–418, 2011.

(b) Cognitive and behavioural studies aimed at establishing theories and models related to people attempting to design software services and service-based applications;

(c) Model-informed approaches or tools aiming to facilitate end-user development and design of software services;

(d) Evaluation and comparative studies of tools, approaches and theoretical models in the area of end user development for services.

The overarching objective of the workshop is to sketch a research agenda on the topic of EUD in service-based computing. We believe the topic is of interest to several streams of research in software engineering, human-computer interaction, services computing. Thus, the purpose of this interdisciplinary workshop is to bring together researchers who have faced the problem of making the SOA paradigm available to end-users, might have some ideas on how to facilitate it, and have experimented with these ideas. We expect to provide a platform for discussion on the potential of SOA for non-technical developers in both professional and personal lives.

Some of the larger questions and issues we want to address during the workshop are the following:

- What are the drivers and obstacles to SOA based EUD?
- What is different about software services compared to conventional software, component-based software and distributed software?
- To what extend are existing methods and tools for supporting end user developers of conventional software applicable to software services?
- What are the suitable principles and approaches for understanding, designing, developing, and evolving software services by people who are not software professionals?
- How can we facilitate uptake of EUD4Services?

The workshop brings together contributions from researchers from a diverse range of interdisciplinary fields, such as human-computer interaction, software engineering, artificial intelligence, computer supported cooperative work and cognitive psychology. To facilitate cross-fertilization between latest research trends in the above areas the workshop has invited research posters in addition to the academic papers.

2 Organization

This workshop is the follow-up to the *EUD4Services* at the *AVI'2010* conference and continues to explore issues raised there[1], ensuring continuity in terms of organisation and PC membership.

The submissions were peer-reviewed for their innovation, relevance to the workshop topics and their potential to generate interesting discussions. Upon acceptance, positions papers were posted on www.EUD4Services.org, our newly established wiki community. Discussions were invited in preparation for the workshop to establish a 'common language' and understanding among the multidisciplinary audience. The

[1] M.F.Costabile, B.E.R. de Ruyter, N.Mehandjiev, P.Mussio: End-user development of software services and applications. Proceedings of the AVI 2010: 403-407.

actual workshop comprises full paper presentations, invited posters and tool demos. An interactive final session is devoted to discussion and summing up.

3 Organizers' Background

Antonella De Angeli is Associate Professor in Human-Computer Interaction at the Department of Information Engineering and Computer Science of the University of Trento. Her research addresses the cognitive, social and cultural consequences of information technology with an emphasis on the application of this knowledge to interaction design. She has a PhD in Experimental Psychology from the University of Trieste where she completed a 2-year post-doctoral research in Applied Cognitive Psychology. She worked as invited researcher at the Oregon Graduate Institute, Loria (Nancy) and IRST (Trento). From 2000 to 2004 she was a senior HCI researcher for NCR Ltd and then joined the University of Manchester. She has published over 100 papers on her HCI research, serves in the editorial board of major HCI journals (including the International Journal of Human-Computer Studies) and regularly sits in the program committee of leading conferences (e.g., DIS, Interact, AVI and CHI2008). She has organised 5 workshops at major International conferences (CHI 2006, Interact 2005-2007, AVI 2008) and acted as Principal Investigator for the University of Manchester in the EU FP7 project ServFace.

Nikolay Mehandjiev is a Reader at the Centre for Service Research of the Manchester Business School. He obtained his PhD for research in user-adaptable office information systems, and organised a number of international workshops on scaling up end user development to organisational context, and on interdisciplinary software engineering research. He has led the University of Manchester's participation in EUD-Net, the European Network of Excellence in End User Development, and has co-edited three special issues of journals devoted to the topic of EUD – two issues of the Journal of Organisational and End User Computing and one of the Communications of ACM. His current research is focused on approaches and models which enable non-technical audience to design dynamic service systems. In this area he leads the University of Manchester's participation in the EU FP7 project SOA4All. Mehandjiev has published two books and more than 100 refereed papers.

Usman Wajid is a Research Associate at Centre for Service Research, Manchester Business School. He obtained his PhD from Manchester Business School. He is interested in end user development in the context of service-based systems to enable collaborative user-driven system and application development. Usman has worked on the EU funded SOA4All project.

Abdallah Namoun is a Research Associate at Centre for Service Research, Manchester Business School, where he also obtained his PhD. His research interests include user interactions with technology, design of visual interfaces, and methods for testing usability. He has worked on models for re-usable interfaces, software re-use, cognitive modelling, usability engineering, end user development, and requirements engineering. Abdallah has worked on ServFace and SOA4All.

Alberto Battocchi is a post-doctoral researcher in Human-Computer Interaction at the Department of Information Engineering and Computer Science of the University of Trento (Italy). His main research interests concern the design and evaluation of innovative technologies for the rehabilitation of people with cognitive disabilities. In 2008, Alberto received his PhD in Cognitive Science and Education by the University of Trento. He has worked as researcher in the Intelligent Interaction and Interfaces research group of Fondazione Bruno Kessler (Italy) and has been visitor at the Centre for Interdisciplinary Applications of Computer Science of the University of Haifa.

4 Program Committee

Alexander Brändle, FHDW, Germany
Jill Cao, Oregon State University, USA
Fabio Casati, University of Trento, Italy
Maria Francesca Costabile, University of Bari, Italy
Joëlle Coutaz, Laboratory of Informatics of Grenoble, France
Florian Daniel, University of Trento, Italy
Scott Fleming, Oregon State University, USA
Steffen Goebel, SAP Research, Germany
Neil Maiden, City University, UK
Maristella Matera, Politecnico di Milano, Italy
Fabio Paternò, CNR-ISTI, Italy
Volkmar Pipek, University of Siegen, Germany
Boris de Ruyter, Philips Research, The Netherlands
Christian Zirpins, KIT, Germany

5 Accepted Papers

Papers are available from the EUD4Services portal (http://EUD4Services.org); their abstracts are included below.

EUD for Semantic Orchestration of Web Services in Task Management System
F. Ariano[1], B.R.Barricelli[1], M.Padula[2], P.L.Scala[2], S.Valtolina[1].
[1]Dipartimento di Informatica e Comunicazione, Università degli Studi di Milano; [2]Istituto per le Tecnologie della Costruzione, Consiglio Nazionale delle Ricerche.

This paper presents a Task Management System based on a Web service architecture on which a network of software environments is developed. The environments are devoted to support end users in performing End-User Development activities and in exploiting their knowledge and expertise about their business processes. TMS network allows end users, who are experts of a specific domain and work together in the same organization, to design a workflow through visual composition of Web services, to visually validate its execution and to execute it at use time. In particular, the implementation of the TMS Editor, i.e. the environment dedicated to the workflow designer, is presented. The TMS Editor is used to transform the task analysis documents prepared by domain experts into a description of the workflow in terms of the components needed and the relationships among them. The components retrieved and used by the workflow designer are Web services that are available in remote or local repositories. The TMS was presented in its first stage of design and development at EUD4Services 2010.

Hardware on End-User Development
A. Alessandrini and A. Rizzo.
University of Siena, Communication Science Department.

In this article we present the design process conducted during the prototyping phase for the Aahrus Design Project (AaDP), an open design workshop and space for schools, students and teachers. Students without any previous competence on programming and hardware assembly constructed a functional prototype during a week-long workshop. A digital bricolage development strategy was adopted, where, end-user development tools as Arduino, and Processing were the main resources, together with web resources as Instructable.com and makezine.com. During the development we encounter several challenges on construction deriving both from hardware and software components. With this contribution, we would give a wider perspective on end user development, which, partially comprehend also hardware as important issues to consider in the design of end-user development tools.

End-User Composition in Mobile Pervasive Environments
J. Floch[1], P. Herrmann[2], M.U. Khan[2], R. Sanders[1], E. Stav[1], and R. Sætre[2].
[1]SINTEF ICT; [2]Norwegian University of Science and Technology (NTNU).

Intelligent objects and devices are becoming part of the environment where people live. The more mobile and pervasive computing becomes, the greater opportunity users potentially have to customize the computing activities that take place around them. For some people the availability of devices and services offers possibilities for tailoring things to exactly what one wants. For others however, this represents a problem: how to manage the complexity? It is neither practical nor economical to use professional software developers for individual tailoring. Thus, we have to provide users with easily operable tools for service composition. The goal of this paper is to highlight the main challenges for a meaningful end-user tool support.

Integration of Services based on the Community Metaphor: Some Guidelines from an Experience of Use
M.P. Locatelli and C.Simone.
University of Milan-Bicocca.

The community metaphor has been used to define a framework (called Community-Aware-MAS, CASMAS) where existing services can be integrated to define a collaborative support of group activities. The paper reports on the outcomes of an experience of use and identifies some guidelines that could help cooperating end-users in building their collaborative application in this technological setting.

Information Systems' Self-Development as a Model of End-User Development in Networked Organizations
M. Roost[1] and G. Piho[2].
[1]Department of Informatics, Tallinn University of Technology;
[2]Clinical and Biomedical Proteomics Group, CRUK, LIMM, University of Leeds.

The term ‘Networked Organization’ is used to describe a variety of new emergent organizational structures such as Virtual and Learning Organizations. Such organizations need evolutionary information systems that are able to survive over time and have built-in support to handle evolution. The development process of such information systems we call as information systems’ self-development. We describe the state-of-the-art of an ongoing research on information systems’ self-development methodology and introduce some main ideas for the (model driven, service and agent oriented) architecture of evolutionary information systems and their development processes. The self development is interpreted as a model of end-user development in the context of networked organizations and their evolutionary information systems.

Social Discovery and Composition of Web Services

A. Maaradji[1,2], H. Hacid[1], R. Skraba[1], A. Lateef[1], J. Daigremont[1], and N. Crespi[2].
[1]Alcatel-Lucent Bell Labs France; [2]Telecom Sud Paris.

In this paper, we propose a new approach for services recommendation to assist services composition in a Mashup environment capturing and analyzing social interactions. This approach uses an implicit social graph inferred from the common composition interests of users. We describe in detail the transformation of users-services interactions into a social graph and a possible means to leverage that graph to derive service recommendation. This proposal was implemented within a platform called SoCo and the experiments show interesting results.

Smart Services in Smart Home Environments

D. Cavone and B. De Carolis.
Dipartimento di Informatica, Università di Bari.

A Smart Home Environment (SHE) aims at supporting people in daily activities by adapting service fruition to their goals. Therefore, it is crucial to map opportunely users goals and services. This problem can be solved by leaving completely the initiative to users, by providing them with interfaces for easy and intuitive service mash-up, or by proactively planning service composition, thus living the initiative to the environment. In this paper we propose an agent-based approach that, on the basis of the recognized situation and user goal, combines services of the physical environment with the net-centric ones. In doing so, it leaves the control to the user by means of an interface that allows controlling the proposed services by accepting or changing the suggested combinations.

Acknowledgment. We would like to thank the organizers of the IS-EUD 2011 conference, for giving us the opportunity to organize our workshop. We would also like to acknowledge the help and professional attitude of the Programme Committee members for their reviews within a short deadline.

DEG: Involving End Users and Domain Experts in Design of Educational Games

Carmelo Ardito[1] and Nikolaos Avouris[2]

[1] Università degli Studi di Bari, Dipartimento di Informatica, via Orabona 4, 70125 Bari, Italy
ardito@di.uniba.it

[2] Human-Computer Interaction Group, University of Patras, Greece
avouris@upatras.gr

Abstract. Designing educational games is an arduous task that requires a multidisciplinary team, whose components must be provided with tools allowing them to actively participate in the creation of such games. This first edition of the DEG Workshop aims at providing researchers interested in this area the possibility to share and discuss their experiences. This workshop is addressed to researchers and practitioners, involved in the design and evaluation of technology-supported games, to discuss their experience in relation to means for involving end users as well as experts in the process, before, during and after the product has been completed. Issues such as how the technology affects the process, in particular in terms of game genres and technologies used (e.g. city games, mobile games, educational games, games on multitouch displays etc.), are examined. Special attention is given to scenarios that affect the expected user experience, measuring factors like pleasure, learning outcome, etc. and the effect of end-user involvement on them.

Keywords: end-user development, meta-design, educational games.

1 Introduction

The latest hardware advances and the quest for successful and innovative learning approaches have led to computer-based edutainment, i.e. computer applications that achieve learning through entertainment [1], [2]. There is evidence that well-designed computer games can meet some of the psychological needs of children and motivate them to learn [3]. The use of mobile devices could expand learning opportunities, freeing the users from the desktop, supporting interaction with learning objects in different ways while exploring a physical environment. Recently, various examples of pervasive games with learning objectives have been reported, to be played indoors [4], [5] or outdoors [6], [7]. They have been developed primarily to support visitors to museums, archaeological parks, historical city centers, etc.

The End-User Development (EUD) perspective acknowledges the importance of involving different experts in the design of effective interactive systems, since they bring different kinds of knowledge, needed in addition to typical software development skills [8], [9]. Of particular importance are the experts of the application domain, while in the case of educational games, education experts as well as HCI

M.F. Costabile et al. (Eds.): IS-EUD 2011, LNCS 6654, pp. 419–424, 2011.

experts are required. Such experts are active participants in the design, development and evaluation of the software [10]. End users are also involved in the design and development of computer games to a great extent: in fact, most modern computer games allow end users to modify many of the determining factors of the user experience, as users can modify the setting, the characters, the environment, the story, behaviors of objects and actors etc. Often they can extend the functionality and modify the rules of the game, even through cheating. So, as a result, the game design goes beyond the end of the typical design process involving the end users. EUD goes beyond participatory design in that it stresses a more active involvement of end users in the different phases of the software life cycle: design, development, evolution. To this aim, they have to be provided with software environments and tools through which they can be actively involved in adapting, modifying or even creating software artifacts [11].

2 Workshop Presentations

The following papers will be presented during the workshop:

A Modular approach for Supporting Multidisciplinary Design Educational Game Experiences. Telmo Zarraonandia, Paloma Diaz and Ignacio Aedo (Universidad Carlos III de Madrid, Spain).

In this paper the authors propose a model that aims at providing a common framework for describing the educational game experience from different perspectives. In order to facilitate the reuse of game design components and the description of new game variants, the elements of the model are organized modularly and so they can spread over different layers and sub-models to support different educational experiences.

Challenges in Designing Domain-Specific Modeling Language for Educational Games. Olga De Troyer and Elien Paret (Vrije Universiteit Brussel, Belgium).

In this paper, the challenges related to the design of domain-specific modeling languages to design educational games are discussed. A domain- specific modeling language uses the vocabulary of the application domain and provides abstractions that make the specifications of solutions easier and more accessible for domain experts and end users. Although such an approach looks promising for involving domain experts and end users in the design process of educational games, several research questions still need to be solved.

Configuring, adapting and evolving software applications supporting visits to cultural heritage sites. Carmelo Ardito, Maria Francesca Costabile, and Adalberto L. Simeone (Università degli Studi di Bari, Italy).

This paper presents an EUD approach for the development of interactive applications for visiting cultural heritage sites. A fundamental role is played by cultural heritage experts, who require personalized design environments which give them the means to further adapt the final applications according to evolving needs of the end users. The approach builds on a theoretical model defining the stakeholders communities participating in the design of such applications, the resources involved and the relations among them. A framework comprising the design environments tailored to the communities defined by the model that is developed is presented.

Content creation by end users for location-sensitive mobile educational games. Christos Sintoris, Dimitrios Raptis, Nikoleta Yiannoutsou (University of Athens, Greece), Sotirios Dimitriou and Nikolaos Avouris (University of Patras, Greece).

One of the main challenges for the broader adoption of location-based mobile games for learning is the process of creating useful content. End-users, e.g. teachers and facilitators who use such games may be actively involved in this process. In this paper the use of social media as tools for collaboratively creating content for location-sensitive mobile educational games is proposed. Experience of using tools for end-user content creation for two location-sensitive mobile games is described. The game design process is presented as the interplay between technology, learning and content with the content generation as a distinct phase of the process. Finally opportunities and limitations of using social media for content-creation are outlined.

Creating large-scale educational games with the Fun in Numbers platform. Irene Mavrommati (Hellenic Open University, Greece), Georgios Mylonas and Ioannis Chatzigiannakis (University of Patras, Greece).

In this paper, the possibilities offered by utilizing pervasive computing technologies, for creating large-scale multiplayer games and interactive installations are presented. Wireless sensor networking hardware is used as the basis for the *Fun in Numbers* software platform. The vision is to enable immersive experiences largely based on the participation of large groups of users, movement and activity in the physical space, providing a generic framework that can be tailored to specific instances. User survey results from a 3-day public exhibition are presented, along with a discussion on the opportunities of such platforms. Based on this, various ways are presented for domain experts or end-users to use such tools to generate experiences suited to their educational/artistic needs.

End User configuration of game elements: Game construction as learning activity. Nikoleta Yiannoutsou (University of Athens, Greece), Christos Sintoris and Nikolaos Avouris (University of Patras, Greece).

End user configuration of game elements is analyzed here from two different perspectives a) as a type of end user involvement in computer game development and b) as a tool for learning. A template is described as computer environment that supports students to construct their own computer based treasure hunt games by manipulating the basic elements of the game. An indicative analysis of the learning process during game construction suggested that game element configuration shaped a rich learning activity that challenged students not only to engage in spatial concept negotiation but also to consider issues related to game design.

Involving Learners and Domain Experts in the Analysis of the Context of Use for the TERENCE Games. Tania di Mascio (University of L'Aquila, Italy), Rosella Gennari (FUB, Bolzano, Italy) and Pierpaolo Vittorini (University of L'Aquila, Italy).

TERENCE is an adaptive learning system for poor comprehenders, that is, children that demonstrate text comprehension difficulties, related to inference-making, despite proficiency in low-level cognitive skills like word reading. Its learning material will be stories and games for its stories. There are several pencil-and-paper interventions by psychologists for improving inference-making skills, and educators have their own methods for supporting poor comprehenders. In order to analyse such interventions and transform them into the smart games of TERENCE, a user-centred design methodology has been adopted. This means, first of all, analysing and specifying the context of use of the system via an ad-hoc approach, consisting of a preparatory preliminary study, followed by field studies. The paper describes this approach and hence the main results relevant for the design of the TERENCE games.

Leveraging the Teachers' Experience to Develop Educational Micro-Games for First Grades Pupils. Raffaele De Amicis, Giuseppe Conti, Gabrio Girardi, Michele Andreolli (Università degli Studi di Trento, Italy), and Silvia Bordin (Fondazione Graphitech, Trento, Italy).

The purpose of this paper is to present the results of a research work focusing on the development of a serious game prototype for early primary school. The project builds on top of experience gathered from teachers involved in an experimental curriculum based on use of game. The game, targeted to pupils aged between 6 and 7, is called the isle of Wii, and it is structured as a container for micro-games designed to promote development of basic skills in mathematics. The game is based on a three-dimensional game engine and it makes use of the WiiMote game controller to promote a playful interactive experience with the game. The paper discusses the implication of educational games in the context of these classes, it illustrates the interaction metaphor developed and shows how this has been essential to ensure a more exciting and fun experience tailored to young pupils.

Game playing and learning in the field: The design and evaluation of a mixed-reality game for an art museum. Konstantinos Mikalef and Konstantinos Chorianopoulos (Ionian University, Greece).

The goal of this work is to explore the effects of a mixed reality game to learning and museum experience. The game content was focused on visual elements, and took place at the Art Gallery, of the municipality of Corfu (Greece). The development of the game was based on QR and quiz software for mobile phones. The only hardware requirement for running the game was a java enabled mobile phone with a camera feature. In addition to the interactive version of the game, a paper-based version of the same game was designed, in order to compare the two with traditional museum visits. Moreover, a control group was employed, which did not play any of the two versions of the game, but had a guided visit of the same art gallery. In this position paper, the motivation, the design of the game, the experimental design, as well as early findings from a field study with fifty-five students are included.

Bending the Rules…and Adding Some New Ones: Legal and Illegal Behaviours of Players in Massively Multiplayer Online Games. Iro Voulgari and Vassilis Komis (University of Patras, Greece).

This paper focuses on the interaction of the players' community in Massively Multiplayer Online Games (MMOGs) with the game environment from the perspective of the rules, and the possible implications of these interactions, for the game design. Two main types of behaviours are identified: the breaking of rules and the development of new ones. The finding is that acceptability of these rules and behaviours are conditioned by both personal criteria and the social context.

3 Workshop Plan

The workshop will be a day long. A keynote statement will be presented first. Then presentations included in the previous section. In the second part, participants will discuss the main points raised and get involved in group design activities and report to the final plenary session. Extended versions of reviewed papers will be considered for publication in an edited volume.

4 Organizers Background

Carmelo Ardito received the PhD in Computer Science in May 2008 at the University of Bari, Italy. Since April 2008 he is research fellow at the Computer Science Department of the University of Bari. He is member of the Interaction, Visualization and Usability (IVU) Lab, coordinated by Prof. Maria Francesca Costabile. His current research interests are in Interaction with mobile and ubiquitous systems, information visualization, usability and user experience, educational pervasive games, end-user development. Dr. Ardito served as Demo Co-Chair in the International Conference IDC 2009 (Interaction Design and Children). He is Local Arrangement Chair of the third International Symposium on End-User Development (IS-EUD 2011). He is member of ACM, ACM SIGCHI and SIGCHI Italy (the Italian Chapter of ACM SIGCHI). He is Working Group Member in the EU COST Action IC0904 “TWINTIDE (ToWards INtegration of Trans-sectorial IT Design and Evaluation)”.

Nikolaos Avouris is a professor of Software Technology and Human-Computer Interaction with the University of Patras, Greece. His main research interest is related with design usable interactive technologies, while in recent years he has been experimenting with user experience with mobile and context-aware applications. Learning technologies have also been the focus of his research in many recent years. In the cross-roads of these two fields is the game-based learning area of research that is the subject of this workshop. Prof Avouris has teaching experience in industry and academia for over 25 years: he was research fellow of the University of Manchester, UK (1983-1984); Assistant Professor of Computer Science in Athens Technical Education College (1985-1986); Scientific officer of the European Commission served at the Joint Research Centre Ispra, Italy (1986-1993); Senior Software Engineer of the Public Power Corporation of Greece (1993-1994). Associate Professor of Software Technology at the Electrical and Computer Engineering Department of the University of Patras (1994- 2001). Full professor of Software Technology and Human-Computer Interaction and leader of the Human-Computer Interaction group since 2001.

5 Program Committee

The Workshop Program Committee that has responsibility for organizing the event and reviewing the submitted papers has the following members:

- Nikolaos Avouris, University of Patras, Greece (coordinator)
- Carmelo Ardito, University of Bari, Italy (coordinator)

- Franca Garzotto, Politecnico di Milano, Italy
- Panos Markopoulos, Technische Universiteit Eindhoven, NL
- Irene Mavromati, Hellenic Open University, Greece
- Sharon Oviatt, Incaa Designs, USA
- Janet Read, University of Central Lancashire, UK
- Maria Roussou, Makebelieve and University of Athens, Greece
- Nikoleta Yiannoutsou, University of Athens, Greece

References

1. Pan, Z.: E-learning and game. Computers & Graphics 30, 1–2 (2006)
2. Roussou, M.: Learning by doing and learning through play: an exploration of interactivity in virtual environments for children. Comput. Entertain. 2, 10 (2004)
3. Garzotto, F.: Investigating the educational effectiveness of multiplayer online games for children, pp. 29–36. ACM, New York (2007)
4. Cabrera, J.S., Frutos, H.M., Stoica, A.G., Avouris, N., Dimitriadis, Y., Fiotakis, G., Liveri, K.D.: Mystery in the museum: collaborative learning activities using handheld devices, pp. 315–318. ACM, New York (2005)
5. Sintoris, C., Stoica, A., Papadimitriou, I., Yiannoutsou, N., Vassilis, K., Avouris, N.: MuseumScrabble: Design of a mobile game for children's interaction with a digitally augmented cultural space. International Journal of Mobile Human Computer Interaction 2 (2010)
6. Ardito, C., Buono, P., Costabile, M.F., Lanzilotti, R., Pederson, T., Piccinno, A.: Experiencing the Past through the Senses: An M-Learning Game at Archaeological Parks. IEEE Multimedia 15, 76–81 (2008)
7. Rogers, Y., Price, S., Randell, C., Fraser, D.S., Weal, M., Fitzpatrick, G.: Ubi-learning integrates indoor and outdoor experiences. Commun. ACM 48, 55–59 (2005)
8. Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A.: Visual Interactive Systems for End-User Development: A Model-Based Design Methodology. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 37, 1029–1046 (2007)
9. Lieberman, H., Paternò, F., Wulf, V. (eds.): End User Development. Springer, Heidelberg (2006)
10. Schuler, D.: Participatory design: principles and practices. L. Erlbaum Associates, Hillsdale (1993)
11. Costabile, M.F., Fogli, D., Lanzilotti, R., Mussio, P., Parasiliti Provenza, L., Piccinno, A.: Advancing End User Development Through Metadesign. In: Clarke, S. (ed.) End User Computing Challenges and Technologies: Emerging Tools and Applications, pp. 143–167. Information Science Reference, Hershey (2008)

Author Index