Lecture Notes of the Institute
for Computer Sciences, Social Informatics
and Telecommunications Engineering 110

David Uhler Khanjan Mehta
Jennifer L. Wong (Eds.)

Mobile Computing, Applications, and Services

4th International Conference, MobiCASE 2012
Seattle, WA, USA, October 11-12, 2012
Revised Selected Papers

Volume Editors

David Uhler
National Diment - Slalom Consulting
Seattle, WA 98104, USA
E-mail: davidu@slalom.com

Khanjan Mehta
The Pennsylvania State University
School of Engineering Design Technology
and Professional Programs (SEDTAPP)
University Park, PA 16802, USA
E-mail: khanjan@engr.psu.edu

Jennifer L. Wong
Stony Brook University, Computer Science
Stony Brook, NY 11794, USA
E-mail: jwong@cs.stonybrook.edu

ISSN 1867-8211 e-ISSN 1867-822X
ISBN 978-3-642-36631-4 e-ISBN 978-3-642-36632-1
DOI 10.1007/978-3-642-36632-1
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013931554

CR Subject Classification (1998):
C.5.3, H.3.4-5, H.4.1-3, H.5.1-3, I.2.10, D.1.5, D.1.m, C.2.1,
C.2.4-5, H.1.2, K.4.1, K.4.4

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

This volume contains papers presented at the 4^{th} International Conference on Mobile Computing, Applications, and Services (MobiCASE) which was held October 11–12, 2012, in Seattle, Washington USA. MobiCASE brought together leading researchers, developers, and professors in the field of mobile technologies. The conference was sponsored by ICST and technically co-sponsored by IEEE Computer Society and Create-Net in association with the European Alliance for Innovation (EAI). MobiCASE was also sponsored by Slalom Consulting.

The aim of MobiCASE conferences is to provide a platform for researchers from academia and industry to advance mobile application and services research, an exciting area that has attracted significant attention from the community in recent years. The first MobiCASE conference, MobiCASE 2009, was held in San Diego, during October 26–29, 2009; the second, MobiCASE 2010, was held in Santa Clara, during October 25–28, 2010; the third, MobiCASE 2011, was held in Los Angeles, during October 24–27, 2011.

Innovators from around the world converged in Seattle to share the latest knowledge on mobile applications and services research.

This year, the conference received 51 submissions and the Program Committee selected 18 papers for oral presentation at the conference, following a highly selective review process. An additional nine papers were accepted for poster presentation.

In addition to the regular papers included in this volume, the conference also featured a keynote speech titled "Musubi: A Novel Phone-to-Phone Social App Platform" by Monica Lam from Stanford University and an industry panel featuring Hoorah! Mobile and iRespond.org. The Technical Program Committee decided to give the Best Paper Award to Xiaochao Yang, Chuang-Wen You, and Andrew Campbell, for the paper "Visage: A Face Interpretation Engine for Smartphone Applications."

I would like to sincerely thank the members of the Program Committee and their referees for their hard work in reviewing the submissions. We are grateful for the generous support of MobiCASE 2012 by Slalom Consulting and to ICST for sponsoring the event. We were honored to be part of this year's MobiCASEand look forward to the continued success of the conference in coming years.

October 2012 Jennifer L. Wong

Organization

Organizing Committee

Conference General Co-chairs

Khanjan Mehta	The Pennsylvania State University, USA
David Uhler	Slalom Consulting

Technical Program Committee (TPC) Chair

Jennifer L. Wong	Stony Brook University, USA

Local Organizing Chair

Jennifer Owens	Slalom Consulting

Demos and Poster Chair

Ralf Cabon	Fraunhofer IESE, Germany

Workshops Co-chairs

Gerard Memmi	Telecom-ParisTech, France
Arjmand Samuel	Microsoft

Industry Chair

Won Jeon	Samsung R&D

Sponsorship Chair

Daniel Maycock	Slalom Consulting

Publicity Co-chairs

Weidong Shi	University of Houston, USA
Aaron Fleishman	

Steering Committee

Martin Griss	Carnegie Mellon University, USA
Thomas Phan	Samsung R&D U.S. Labs, USA
Petros Zerfos	IBM T.J. Watson Research Center, USA
Imrich Chlamtac	Create-Net, Italy

Program Committee

Davide Brunelli	University of Trento, Italy
Yi-Fan Chen	Old Dominion University, USA
Amadou Daffe	Coders4Africa
Guangbin Fan	Huawei USA
Roozbeh Jafari	UT Dallas, USA
Matthew Kam	Carnegie Mellon University, USA
Du Li	Ericsson Research
Ernestina Menasalvas Ruiz	Universidad Politécnica de Madrid, Spain
April Mitchell	HP Labs
Ani Nahapetian	California State University, Northridge, USA
Amit Nanavati	IBM India
Elin Pedersen	Google
Irene Petrick	Penn State University, USA
Arno Puder	San Francisco State University, USA
Lori Scarlatos	Stony Brook University, USA
Christelle Scharff	Pace University, USA
Norbert Seyff	University of Zurich, Switzerland
Pritpal Singh	Villanova University, USA
Revi Sterling	University of Colorodo, USA
Bill Thies	Microsoft
Conrad Tucker	Penn State University, USA
Cecilia Wandiga	Global Ectropy

Table of Contents

Mobile Application Development

Multi-Dimensional Interactions

System Support and Architecture

Mobile Applications

Mobile Services

Poster Session

Erratum

mConcAppt – A Method for the Conception of Mobile Business Applications

Steffen Hess, Felix Kiefer, Ralf Carbon, and Andreas Maier

Fraunhofer IESE
Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
{steffen.hess,felix.kiefer,ralf.carbon,andreas.maier}
@iese.fraunhofer.de

Abstract. Mobile business applications (mobile business apps) bear huge potentials for increased work productivity, work comfort, and even sales if they are of high quality. Usability and user experience, in particular, are among the key quality attributes. The high quality requirements of mobile business apps require them to be thoroughly engineered. Unfortunately, today's software engineering approaches are often too heavy-weight to allow developing high-quality mobile business apps in the context of mobile projects, which often face small budgets, extremely limited effort, and short time-to-market requirements. This paper presents mConcAppt, a user-centered and lightweight conception method for mobile business apps. It provides guidance for requirements engineering and interaction design for mobile business apps and provides interfaces to other activities, such as visual design, architectural design, implementation, and testing. The adoption of mConcAppt in various industrial contexts indicates that it enables organizations to elaborate a concept for a high-quality mobile business app in 2-4 weeks.

Keywords: Mobile Business Applications, User Experience, Interaction Design Method, Requirements Engineering, User-Centered Design.

1 Introduction

More and more often, organizations try to exploit the potential of mobile applications (mobile apps) for their business. They do so to increase the work efficiency and work comfort of their employees (B2B scenario) and enhance sales by offering mobile services to their customers (B2C scenario) by means of mobile business apps. There is a huge potential in using not only mobile apps for personal information management (email, calendar, tasks) or general purpose apps, for instance to take notes, but also highly specialized mobile business apps built to support employees' specific work tasks or mobile customers. Both in the B2B and the B2C scenario, mobile apps must guarantee high quality, as they are used in potentially business-critical situations.

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 1–20, 2013.

Organizations aiming at developing mobile business apps and especially the involved requirements engineers and interaction designers are faced with the following challenges (see Fig.1).

High usability and user experience - Usability and user experience, in particular, are essential quality attributes of mobile business apps [11]. Users of mobile business apps expect to immediately put them to productive use. Typically, users will neither read manuals explaining the usage of a mobile app nor frequently use a help system or call support hotlines. They expect a mobile business app with a unique user experience that is tailored to their device and the underlying mobile device platform. This means, for instance, that available sensor data is leveraged wherever it makes sense, that multi-touch gestures can be used, that the mobile business app adheres to the user interface (UI) guidelines of the device platform, and that the app is well integrated with other apps of the respective ecosystem. In the absence of high usability and high user experience, mobile business apps will not be used frequently and will not generate the expected effects; they will get de-installed again quickly, and if they are publicly available, bad reviews will be posted in app stores.

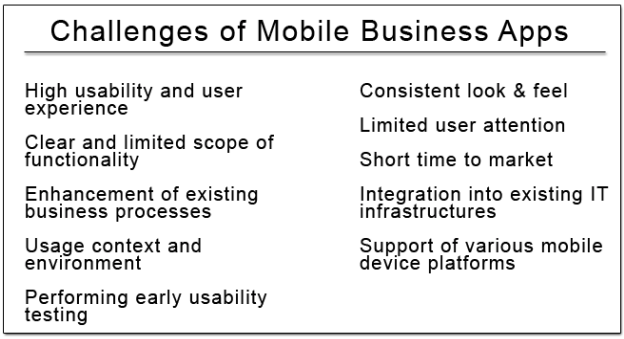


Fig. 1. Challenges of Mobile Business Apps

Clear and limited scope of functionality - Another key success factor of mobile apps we observed in practice is a clear and limited scope of functionality. Mobile users only perform specific tasks on the go. Therefore, a mobile business app should support its user in a small set of clearly defined tasks [16] and should not overwhelm the user with features supporting an overall workflow the way many stationary information systems do [3]. Hence, re-building complete feature sets of desktop applications on mobile devices is not appropriate. A limited and well-grounded set of functionalities of mobile business apps is crucial for their success and acceptance by users.

Enhancement of existing business processes – In organizational environments, existing business processes that bear the potential of being supported by mobile devices have to be identified first [23]. This potential identification must be directly interwoven with the conception of interaction designs to indicate possible mobile support opportunities efficiently. Identifying potentials does not only mean adding the

mobile device to the existing business process; one must also focus on improving the existing process in an intelligent way, which might also change the existing process in general.

Usage context and environment – Environmental challenges are very broad. Besides temperature, light conditions, noise, distraction, mobility of the user, cognitive and physiological constraints of the user [19],[21], the actual usage context, in particular, has to be defined for mobile business apps. In contrast to traditional mobile apps, mobile business apps rely on a concrete organizational task, respectively business process, and therefore the actual usage context can be specified more precisely than "almost anywhere", as it is for traditional apps. In addition, the location awareness of the device and its consideration during conception constitutes a major challenge.

Performing early usability testing – Regarding usability testing, a major challenge for mobile business apps is related to performing tests in the actual usage context with real end users [8]. Usually, access to end users (e.g., pilots) is limited and the usage context often involves customers of the end user (e.g., passengers) who cannot be involved in the process for various reasons (e.g., confidentiality, availability).

Consistent look and feel – Regarding the overall portfolio of business applications used, the mobile business app has to fit into that portfolio to create a consistent look & feel for the end user. This also needs to address legacy software systems and gets even more challenging if more devices and operating systems are added to the portfolio (e.g., smartphone apps, tablet apps, desktop applications, applications for customized systems, etc.).

Limited user attention – Users of mobile business apps often have limited attention for the actual interaction they are performing with the device. In general, this leads to the challenge that the conception of mobile business apps has to consider whether the app will be used during primary tasks (e.g., access information on the tablet while traveling on the train) or secondary tasks (e.g., access information on the smartphone while talking to a customer).

Besides such specific challenges, there also exist more general challenges for the overall development organization (including requirements engineers and interaction designers). These challenges comprise, for instance:

Short time to market – Once it has been decided to bring a mobile app into the market, it has to be realized and delivered quickly. Otherwise, business opportunities can get lost, for instance, if customers buy competitor products that are available in the respective app stores earlier (specifically relevant for the B2C scenario).

Integration into existing IT infrastructures – Mobile business apps rely on data and services from backend systems, which are typically part of the already existing IT infrastructure of an organization [21]. This often requires modifications of backend services or even new backend services that specifically fulfill the requirements of mobile clients.

Support of various mobile device platforms – Particularly if mobile business apps are built for end customers, various mobile device platforms like Apple's iOS, Google's

Android, or Microsoft's Windows Phone must typically be supported. Even if a technology supporting multiple mobile platforms is used, the users of each platform expect a native user experience, i.e., the mobile business app must feel like a native iOS, Android, or Windows Phone app.

Limited hardware resources - Mobile devices bear some huge constraints, especially regarding the usability of mobile apps (e.g., small screen sizes, limited input facilities, limited device capacity, limited power supply) [1],[15].

These challenges reveal that mobile business apps must be thoroughly engineered to satisfy users, customers, and all other involved stakeholders. Unfortunately, we have observed in various industrial settings that mobile business apps are poorly engineered and do not provide appropriate quality. Organizations developing mobile business apps do not adopt proven software engineering methodologies in the development of mobile business apps, as they appear too heavyweight for developing a piece of software that is relatively small compared to desktop products or overall IT infrastructures. Hence, mobile software engineering has to provide tailored engineering solutions for mobile business apps.

This paper introduces mConcAppt (used as an abbreviation for "method for the conception of mobile business apps"), a method for the conception of mobile business apps. Here, conception subsumes requirements engineering as well as UI & interaction design. This integrated view of the two disciplines of software engineering originates from the TORE approach [2]. The method is user-centered to allow striving for high usability and user experience and to develop apps that fit the needs of its users regarding their scope of functionality. It is lightweight to support organizations in efficiently developing mobile business apps and meet their effort and time-to-market constraints. The conception method is interrelated with other activities in the development cycle. It has interfaces to business analysis, visual design, architectural design, implementation, and testing.

The remainder of this paper is structured as follows. Section 2 provides an overview of related work. In Section 3, we introduce the mConcAppt method illustrated on an example project. Section 4 describes lessons learned during the application of the method in industry projects. Section 5 concludes the paper and gives an outlook on future work.

2 Related Work

Without a doubt, usability and user experience are one of the major challenges for mobile business apps. High usability and user experience is required to gain acceptance by the targeted end user group. Thus, usability and user experience are mandatory success factors for a mobile business app. To guarantee these success factors, user-centered design processes for software design and development are common practice. However, only few approaches exist today that shift their focus from a general user-centered approach towards a user-centered approach for mobile apps. A number of scientists also highlight the need for specific mobile software development methods due to the specific challenges in the mobile context (e.g., usage

of integrated sensors, native, hybrid, and web application development, fast changing surroundings, hard-to-perform evaluations)[1], [7], [24].

Mayhew describes a holistic approach to a user-centered iterative design process, the "Usability Engineering Lifecycle" [20]. It is divided into three phases: requirements analysis, design/testing/development, and installation. The requirements analysis includes user research, task analysis, usability goal settings, and general design descriptions. In the second phase (design/testing/development), the previously elicited requirements are refined and afterwards designed and evaluated in three different states of maturity. The third and final phase (installation) completes the approach with the installation of the product and uses the user feedback occurring after some time to enhance the design, which will then be incorporated into the next releases of the product. The usability lifecycle is a very detailed description of a general approach to usability engineering activities during software development. The mConcAppt approach is oriented towards the usability lifecycle but with a narrower focus on the challenges of mobile business app development. To address these, the single phases of mConcAppt and the resulting artifacts are tailored to the needs of mobile app development. Similar to the usability lifecycle, several other approaches exist (e.g.,[10]) that cover the complete lifecycle of the product with a start-to-finish method. We found that these methods constitute a common ground for interaction design but are too general to be applicable, especially regarding the challenges of mobile business apps. mConcAppt overcomes this gap by tailoring existing concepts from those methods in an applicable manner to allow them to be used under the given challenges of mobile business apps.

Grill et al. describe a pattern approach to mobile interaction design [12]. The approach starts with the elicitation of requirements with a focus on the analysis of the mobile environment. In this case, the term mobile environment means the environment where a mobile scenario takes place. The mobile technology, the users, as well as data and information are parameters for the mobile environment. After being elicited, the parameters are used in a second step as the basis for the upcoming interaction design process (IxD-Process). The IxD-Process is iterative and consists of building numerous design drafts followed by a formative evaluation. Beneath common tools for interaction design like personas and scenarios, the interaction designer has tool-based access to a library of mobile interaction pattern. Thus, the interaction designer is able to use common practices in the field of mobile interaction design to generate a design solution. The evaluation of the interaction designs created in this tool has shown that especially expert users of mobile devices have some trouble with the interaction because of their concrete experiences and expectations regarding interaction with a mobile device. Our experience shows that the use of patterns for mobile interaction design may lead to underestimating the importance of the actual usage context because of the use of general solutions for known design problems. To avoid this, the mConcAppt method aims for direct communication between the interaction designer and the user of the app in order to address the context of use during the whole process and at every step where design decisions are made. This aspect is also not addressed sufficiently in the pattern-based interaction design method.

De Sá and Carrico describe in [7] lessons learned from the design processes of several mobile apps. They emphasize three stages of app development that lead to particular challenges in terms of mobile devices, namely data gathering, prototyping, and evaluation. In [8], they describe their own design methodology for mobile devices. This methodology contains three phases, pervasive data gathering, early stage prototyping, and mobile evaluation. The first phase is used to gather data and elicit requirements in order to sufficiently address the context of use. The early stage prototyping phase is used to develop prototypes based on the gathered information. These prototypes are developed for evaluation in-situ to address the challenge of context of use again. Such evaluations take place during the last phase of this methodology. To keep the conditions as realistic as possible, the evaluation is conducted in the previously elicited context of use. The authors do not provide techniques or a start-to-end method in their approach. Furthermore, they complete the description of every phase with resulting guidelines from their experience with various development processes. These guidelines focus on the challenges of mobile apps. Thus, this methodology and these guidelines are promising, but they lack concrete guidance and templates, which are provided by mConcAppt. Thus, the mConcAppt approach can also be applied by mobile interaction designers without much experience. Another aspect is that the main focus of this methodology is the in-situ design, which tackles the context of use challenge. In general, it is a very good practice to elicit requirements for mobile apps and evaluate them in the actual context of use. But we found that limited budgets for mobile apps often hinder such field studies. In addition, mConcAppt could be easily enhanced with requirements elicitation techniques such as contextual inquiries to gather information prior to the requirements workshop. However, this is not mandatory for the achievement of successful interaction design by using the mConcAppt approach.

Newer approaches (e.g., [5],[9],[22]) often provide detailed guidance on how to build an app for a certain mobile device or operating system. Unfortunately, they are often not tailored to the specific needs of mobile business apps and do not provide any concrete method for developing an interaction design.

3 mConcAppt Method and Example Project

The mConcAppt method combines requirements engineering and UI & interaction design activities for mobile business apps and produces a so-called interaction concept as a basis for implementation and further activities. This section starts with an overview of the mConcAppt method and then describes each phase of the method in detail.

3.1 Overview

Regarding our user-centered approach, the mConcAppt method is in the center of the overall app development process, acting as a mediator between all activities. Requirements engineering and UI & interaction design activities must be closely interwoven with other engineering activities like architectural design to ensure that the required usability and user experience can be supported by the architecture and to

address trade-offs with other quality attributes besides usability and user experience, which are of course important as well. By defining clear communication interfaces between the different software engineering activities, mConcAppt acts as driver for engineering mobile business apps with high overall quality [13].

Upstream activities (e.g., generating the general app idea, resolving organizational issues, and clarifying responsibilities) comprise all activities that take place prior to the actual app development activities. Downstream activities (e.g., final app development, app distribution) comprise activities that take place after the final interaction concept is delivered. This paper focuses on the description of the mConcAppt method and its phases as shown in Fig. 2. The app conception as a whole is an iterative process. We describe in detail every activity to be performed in one complete iteration of the method.

The initial phase of the mConcAppt method, called *Elicit Requirements,* comprises the gathering of requirements that are especially relevant for the interaction design. The phase *Specify Interaction Design* comprises the actual development of the interaction concept in tight collaboration with responsible persons from other software engineering disciplines. To support this collaboration, the method comprises a *Communicate Interaction Design* phase dedicated to showing how the evolution steps of the interaction concept should be communicated to the roles involved in the app development. The rationale for this communication is to get early feedback from other involved roles and to enable the project stakeholders to perform their activities as early as possible. In this case, the architect and the business analyst are the most important information sources regarding the interaction concept because they provide much information and many constraints that might be considered in the concept. Only after this internal communication and the approval by the project management is the interaction design validated with actual end users in the *Validate Interaction Design* phase.

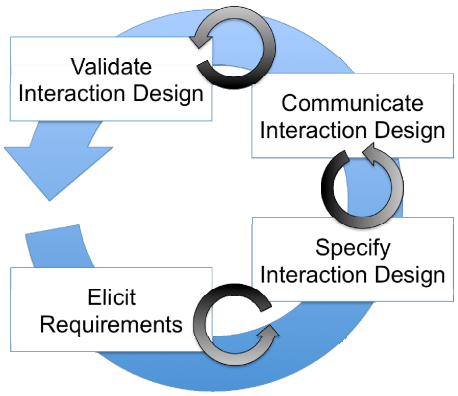


Fig. 2. mConcAppt Overview

The mConcAppt method has been applied in several industry settings and refined based on the practical experiences made there. In the following, the method is described using examples or excerpts of one of our in-house app engineering projects. The mobile business app developed in this project aims to assist employees in

tracking information required for travel expense reports while on a business trip and in creating travel expense reports after a business trip. It has been developed for the iPhone. The app is intended to be used as a showcase project in exhibitions to illustrate mobile software engineering.

3.2 Elicit Requirements

The *Elicit Requirements* phase comprises all requirements engineering activities that have to be performed after the first management decision has been made regarding a specific app. Fig. 3 shows the sequence of activities in the first iteration of this phase and the time typically required for each activity based on our experiences gained when applying the method in industry projects. The aim of the activities of this phase is primarily to prepare and conduct a requirements elicitation workshop and to document the elicited results.

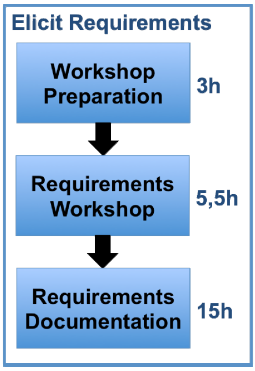


Fig. 3. mConcAppt Elicit Requirements Phase

The workshop preparation subsumes all activities that are necessary prior to conducting the workshop. These activities include creating a workshop agenda, eliciting initial information, and selecting and inviting the participants. The workshop agenda should be aligned with the intended outcomes (see Fig. 4) and allocate enough time for each step. Each step shown in Fig. 4 should be an explicit item on the agenda.

The participants should be selected following Table 1. In particular, the selection of appropriate (lead) users is essential for a successful workshop. The participants should be invited at least one week in advance.

In addition, the workshop organizer has to make sure that the required material for the workshop is prepared (flip charts, beamer, pencil, cards) and that the workshop room is set up to give participants a pleasant environment. Fig. 4 shows an overview of the steps performed during the workshop and the resulting artifacts. The documentation of the workshop results in the form of artifacts is supported by templates provided by mConcAppt.

In our example project, we identified six different stakeholder groups (company employee, developer, travel management, exhibition visitor, company steering committee, and business area manager). We documented their roles and goals and described them using a template. After identifying the company employee as the main target user group, a user persona [6] was developed. The chosen persona should be an instantiation of the user group that represents the most common user (e.g., the 80% user) in the best possible way. The persona describes various goals and behavior patterns of potential users. It also encapsulates and explains critical behavior data in a way that other stakeholders can understand, remember, and relate to. The persona is created during the workshop directly in collaboration with the participants of the workshop, usually in combination with factors that describe the general usage context, especially existing hardware and software experiences. During the remainder of the workshop and throughout the entire development process, this persona will represent the user in upcoming discussions. In our example, the persona describes a team leader in our company. The role of a team leader was selected because the estimated time a team leader spends on business trips is about 40% of his/her total work time. This makes the team leader one of the most important and representative users of the app.

Table 1. Workshop Participants

Participant (Quantity)	**Description and role in the workshop**
Interaction Designer/ Requirements Expert (1-2)	Workshop conception and preparation; moderator of the workshop
(Lead) User (2-4)	Gives input from a user's perspective. The lead user approach should be chosen if no real users are available or if the system offers high innovation potential. "Lead users face needs that will be general in a marketplace – but face them months or years before the bulk of that marketplace encounters them, and lead users are positioned to benefit significantly by obtaining a solution to those needs." [14]
Project Manager (1)	Gives input from the project management perspective. This role is filled by the internal decision maker and the person responsible for the project.
Business Analyst	Takes the minutes of the workshop (the interaction designer is usually not able to make complete notes about the given input because of his moderating tasks); input from business perspective with regard to feasibility can be integrated directly into the minutes to prevent interrupting idea creativity during the workshop.
Customer (1)	Gives input and constraints from a customer perspective. This role is occupied by the decision maker of the customer. This is usually the responsible project management person on the customer's side. Usually this person is not a representative user of the system, although he or she often thinks that he/she knows what the users want.

After the elicitation of the persona and the stakeholders, the as-is-situation including current problems has to be elicited for the tasks that are to be supported by the mobile business app to be developed. In our example, we analyzed the as-is situation of tracking information while on a business trip and creating travel expense reports after returning from the trip. The analysis of the elicited as-is situations revealed several problems that could be tackled with the app:

- The travel expense report requires a lot of information (e.g. locations, dates, distances driven, and mileage) that is not captured during the business trip. In most cases, the traveler forgets to write it down or cannot do so because of a lack of time.
- Often it is not possible to fill out a travel expense report immediately after returning (daily business). This results in a high cognitive load because all information needs to be remembered until the report is created.
- It takes a lot of time to fill out travel expense reports.
- Even business trips without any costs need to be reported.
- Receipts and vouchers from costs incurred during a trip have to be glued on a piece of paper and need to be scanned to digitalize them for further organizational work.
- Each employee fills out travel expense reports in their own way (e.g., crossing out unused parts of the report). This causes additional organizational work in the administration department.

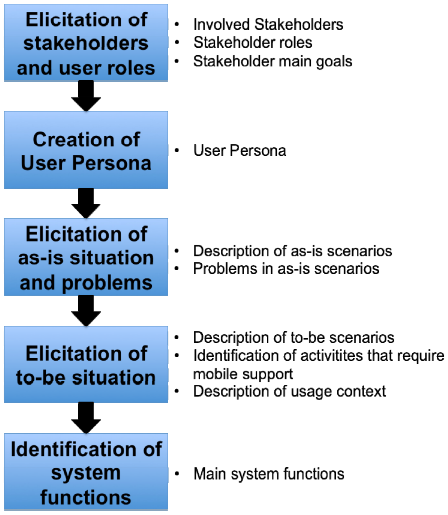


Fig. 4. Steps performed during Requirements Workshop

In order to solve these problems, the to-be-situation was elicited in the workshop. Typically, employees are carrying a mobile device with them during a business trip, which makes it the ideal platform to support precise tracking of business trips. It is, for instance, required to track the exact time when and where (home or office) the business trip starts, when the business appointment starts, etc. The user can track such events of a business trip in real-time, for instance, by interacting with the mobile device when the trip starts. But even better, a mobile business app could recognize such events including locations and start times automatically and propose them to the user. This is possible by aggregating sensor data from the device and up-front knowledge about the respective business trip, for instance from the itinerary. The proposals of the app can be accepted or corrected by the user at any time during the business trip or even afterwards.

Table 2. As-Is and To-Be Scenario Template

Item	Description
Context	Context that leads to the actual task.
Precondition	Precondition for task performance
Step 1-N	Steps that are performed including problems occurring per step.
Postcondition	State that is achieved after the scenario.

In addition to tracking events like the start time of a trip, the mobile app can enable the user to collect artifacts (e.g., tickets and vouchers) during the business trip that are required afterwards for travel expense reporting. If the user uses public transportation and buys a ticket, a picture of the ticket can be taken and added to the travel documentation.

Both the as-is situation and the to-be situation are described using the template shown in Table 2. In addition, steps of the to-be situation that need mobile assistance as well as steps that do not necessarily need to be performed with the support of a mobile business app are identified to get a first indication regarding the mobility potential of the overall process.

The identification of the main system functions completes the workshop. These system functions are based on the previously elicited to-be situations. In addition, exchanged data is identified with the support of the business analyst, who can provide exact information about exchanged data and data formats based on this initial elicitation. The main system functions (e.g., time tracking and collection of travel artifacts) represent the core functionality of the system that has to be designed in the first iteration. During the workshop, it is sufficient to simply name the basic system functions to define the scope of the app.

The requirements documentation processes information gathered during the workshop in a way that allows other involved project stakeholders (see Table 1) to understand the elicited information and design decisions. The workshop documentation is part of the interaction concept description and the basis for all upcoming activities. It can be seen as a first draft version of the concept and documents all elicited information in a structure similar to the workshop agenda. It is a lightweight documentation, focusing on the information that is needed for further steps. mConcAppt also provides templates for the structure of the document. This temporary document can be seen as the first version of the interaction concept. During further iteration cycles of mConcAppt, requirements are elicited as output from the phases *Communicate Interaction Design* and *Validate Interaction Design* and integrated directly into the interaction concept. This workshop is only performed once in the initial phase.

Performing this initial interaction design requirements elicitation phase took 23.5 hours in total. Detailed time estimations for particular phases are shown in Fig. 3.

3.3 Specify Interaction Design

The specification of the interaction design (see Fig. 5) comprises the construction of the actual interaction concept, without consideration of the underlying backend

system in the first iteration. The mobile business app is specified based on information elicited during the initial *Elicit Requirements* phase. First, the key functionalities are derived from the to-be situations. These key functionalities form the starting point of the app and should be represented by the first specified interaction cases. During the specification of the interaction cases, the flow of interaction cases is assembled step by step. After this specification, wireframes are created based on the interaction cases. Analogous to the flow of interaction cases, the actual screen flow is assembled.

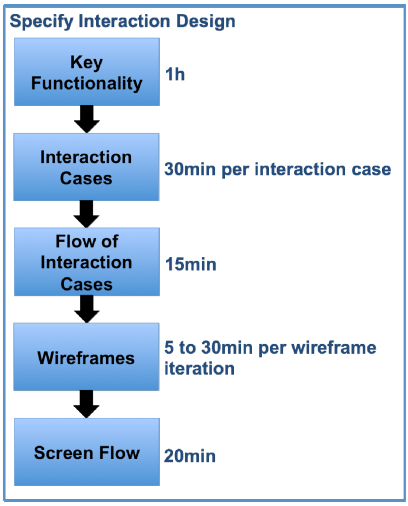


Fig. 5. mConcAppt Specify Interaction Design Phase

In our example project, we took the to-be situation and the main system functions derived from the workshop and identified key functionalities. Key functionalities are tasks that really have to be performed in a mobile way in the given context. There is always a huge risk to come up with too many functionalities based on existing desktop or legacy systems. Ask yourself: "What must really be done on the mobile device?" In our particular case, the key functionalities were limited to time tracking, artifact tracking, and information support for the traveler during a business trip. We explicitly decided not to provide functionality to edit the travel expense report on the mobile device, as the user cannot easily accomplish this on a smartphone. Based on the given information, seven main interaction cases were identified and described, similar to the example shown in Table 3, which is very similar to use case descriptions (e.g., [17]) but focuses only on information needed to specify the interaction concept. Additionally, the usage context is described in detail for each interaction case. According to agile development processes, one interaction case comprises several user stories that are related to a certain task. Usually, exactly one human action and one system action (see Table 3) should form a user story.

When creating the interaction cases, no concrete design solutions are developed; rather, the concept of user interaction is shown. Fig. 6 shows an excerpt of the flow of interaction cases for this particular example project. Arrows are used to indicate the

user flow between different interaction cases and swim lanes might be used to indicate logically divided app contents. All interaction cases are shown in a graphical representation, which allows getting a quick overview of the general structure and function range of the app. Based on the flow of interaction cases, the final decision about the scope of the app is made by the interaction designer in conjunction with the project management.

Table 3. Interaction Case Description Example

Item	Description
ID	IC2: Track Time
Usage Context	A business trip for multiple days. Meeting an industry partner in Leipzig. The user is on his way to the hotel and uses the device while walking.
System Action 1	The system recognizes that the user arrives at his destination and notifies him that the time of arrival is tracked (via notification center).
Human Action 1	The user taps the notification to directly open the app.
System Action 2	The system opens the app and immediately shows the current trip itinerary and proposed time.
Human Action 2	The user confirms the proposed time.
System Action 3	The system provides feedback about the confirmation to the user.
Human Action 3	The user closes the app.
Postcondition	The arrival time is persistently stored.

Using the interaction cases and the flow of interaction cases as a basis, the actual wireframes are created. In first iterations, wireframes are created using paper and pencil. We found that this is the quickest way to create several versions of a screen and to try out different solutions. To guarantee traceability, we use a wireframe template that shows screen identifier, name, version of the wireframe, and date of production. Based on these first paper prototypes, prototypes in other sketching tools (e.g., Balsamiq, PowerPoint, Photoshop) might be created depending on the designers' preference, experience, and time constraints. Fig. 7 shows several versions of the same screen to demonstrate how the prototype evolves in iterations from initial wireframe sketches to the final app design.

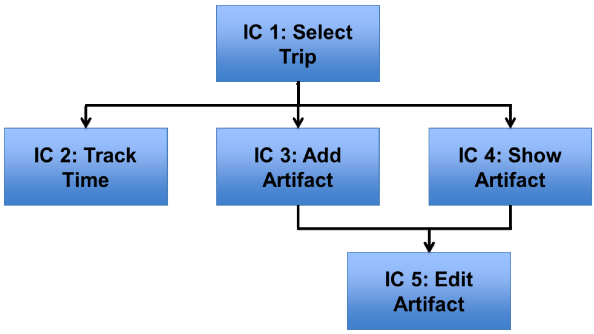


Fig. 6. Flow of Interaction Cases (excerpt)

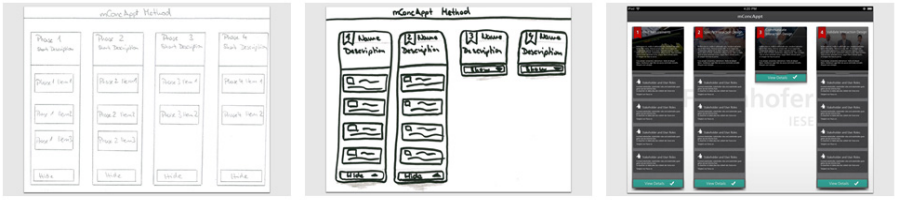


Fig. 7. Evolution stages from first paper mockup to final app

While creating wireframes, the screen flow is assembled. In the interaction case example of "track time" (see Table 3), the simple screen flow only consists of two screens. Typically, the complete screen flow is more complex and represents a holistic view of the complete navigation tree of the interaction concept. Arrows show concrete navigation possibilities, while text along the arrows provides a description of the intended screen transitions. The results of the steps *Elicit Requirements* and *Specify Interaction Design* represent the first final version of the interaction concept. Performing this phase in the given example project took ten hours in the first iteration; time estimates for individual steps are given in Fig. 5.

3.4 Communicate Interaction Design

In this phase, the interaction design is communicated to the other project stakeholders (see Fig. 9). We found that a video explaining the interaction concept is a helpful instrument for communicating it to the other stakeholders and propose this as a best practice for sharing the interaction concept in distributed project teams. Nevertheless, producing a video is an optional step in mConcAppt, especially if the team is not locally distributed. The focus is on exchanging information between all stakeholders and gaining their feedback, which might be transferred to requirements for the interaction design.

To produce the video, the interaction designer must first write a presentation script. Based on this script, snippets are recorded for each interaction case and a complete movie is assembled using these snippets. Once the complete movie is rendered, the interaction designer can distribute the video to the whole project team.

The presentation script is assembled based on the interaction cases and wireframes. The presentation script supports especially inexperienced interaction designers by creating a structured video and is an important part of the interaction design itself, as it documents ideas and reasons for design decisions in a way that allows other project stakeholders to get the story behind the concept and make decisions based on the script, the video, and the overall interaction concept description. It is also important to describe design decisions that are not seen on the actual wireframe at first glance (e.g., screen transitions, gestures, required input feedback, and fancy stuff). Experienced interaction designers might skip this step by performing the activities described above on the fly while recording. Nevertheless, documentation is important for further steps during the development process.

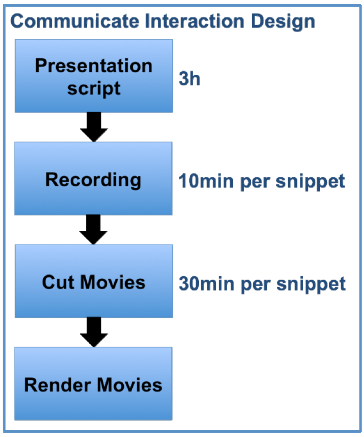


Fig. 8. mConcAppt Communicate Interaction Design Phase

In our example project, we recorded seven snippets according to the given interaction cases. We used a fixed-mounted camera and a pencil to point to elements. The overall length of a snippet was no longer than five minutes and the total video was twenty minutes in length. To prepare for the cutting and assembling of the final video, a template for headlines and overlays that is reusable in other projects is used. Sketches are scanned and/or imported into the authoring software (e.g., iMovie) to show the corresponding wireframe in full screen mode before explaining it. Using this practice, other stakeholder groups are able to easily discuss the actual version of the interaction concept and give feedback on it as well as make decisions based on it. Creating the interaction video for the example project took eight hours in total.

3.5 Validate Interaction Design

As shown in Fig. 2, a real user test is done after the first complete iteration of the method. Fig. 10 shows an overview of the activities during the Validate Interaction Design phase. Based on the interaction cases, concrete usage scenarios [4] form the basis for conducting the test. In addition, a clickable prototype can be created easily using presentation software for non-linear presentations on the mobile device (e.g., Presentation Link). This approach requires marginally more time than applying Wizard-of-Oz testing [18] with paper prototypes, but enables us to perform a user review on the actual end device in the concrete usage environment, which we think is mandatory especially for mobile business apps.

Since the focus was on a small set of features in our example project, user testing was possible very early. We used an interactive on-device prototype with clickable scanned wireframes. The user review was done using a scenario-based approach and we abstained from recording the review. During the next two weeks, the prototype was improved based on the findings and evaluated two more times. For each user review, we chose three to four users.

In the following section, major lessons learned from adopting the mConcAppt method in projects with various industrial customers are presented.

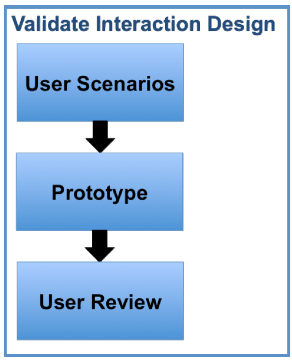


Fig. 9. mConcAppt Validate Interaction Design Phase

4 Lessons Learned

4.1 Elicit Requirements

One of the major lessons learned while using the mConcAppt method is that only one requirements workshop is needed to execute the further activities. However, it is mandatory that all roles described in Table 1 take part in the workshop. Leaving out one of the roles would lead to time-consuming rework and result in lower return on investment. Furthermore, the workshop has to be prepared optimally – information elicited in advance (e.g., observing the users in their as-is situation) increases the quality of the intended outcome. During the workshop, the competence of the moderator significantly influences the quality of the results. The moderator has to adapt to the customer problems and should have a basic understanding of the technical capabilities. Users participating in the workshop often do not discover the potential for mobile support in general, and especially the technical capabilities of the devices are unknown. Potential users of mobile business apps are often constricted by existing workflows, which indeed represent an obstacle to better mobile support.

4.2 Specify Interaction Design

Regarding the specification of the interaction design we learned that it is mandatory that the interaction designer himself is a user of the addressed device class and operating system. To provide high-quality results, it is not sufficient to know operating system programming guidelines. When starting to design the mobile business app, it is a key influencing factor to decide which functionality has to be realized in the backend system and which functionality has to be on the mobile device. Early communication of ideas and sketches across colleagues increases the quality of the results. We found that the interaction designer, who is usually just one person in the case of mobile business apps, needs to step out of his sandbox and

obtain quick informal feedback from colleagues or other available persons. While communicating iterations to other project stakeholders, we observed that there is a large potential conflict area between interaction design and architecture. Software architects tend to refuse ideas and concepts due to technical reasons (e.g., enabling live search, which might be a large boost to user experience but requires significant resources in the backend). Especially user experience enabler and architecture are influencing each other significantly. Regarding mobile business apps, the frontend, which is usually represented by an app, strongly depends on the performance of the backend system. This dependency may have a major impact on the interaction design and creative solutions might be necessary. Project management should be aware of these special circumstances and solve conflicts regarding the overall goals of the project as well as provide prioritization between system capabilities and financial budget.

4.3 Communicate Interaction Design

In the interaction design communications phase, we introduced video and textual communication in combination as the preferred approach, since product teams are often distributed both spatially and temporally. While face-to-face communication should be preferred whenever possible, insisting on lightweight documentation as described will increase the quality of the product.

4.4 Validate Interaction Design

A user review should be performed with four users. For further iterations, the user review team should be composed of two users who took part in previous activities and two users who were not part of previous activities. This composition allowed us to trace whether previously discussed issues had been addressed in an applicable way and discover new issues that might not have been discovered by the users who had taken part in the creation of the concept. Performing the user review on the actual device in the given usage context leads to more reliable feedback than performing tests in a clinical environment using paper mockups.

4.5 General Lessons Learned

It is not beneficial if one person takes over different roles during the workshop because conflicting interests cannot be identified and elaborated. Furthermore, it is often laborious to convince users that they do not have to be able to perform every activity that can be performed with a legacy stationary device while being mobile. The mobile device usually serves as an additional device that supports especially those activities that need to be performed while being mobile.

The complete development of an interaction concept using mConcAppt ordinarily takes two to four weeks, and three iterations were usually found to be sufficient. Concerning the effort and times estimated in the description of mConcAppt, we have to admit that times correlate with the experience of the interaction designer, creative steps cannot really be projected, coordination and communication between the

different development activities still require too much time, and mobile business apps have varying scopes and challenges.

5 Conclusion and Future Work

The mConcAppt method guides its users in performing requirements engineering and UI & interaction design activities for mobile business apps. It aims at addressing the challenges for the engineering of mobile business apps mentioned in the introduction. Thus, *high usability and user experience* are addressed by the overall engineering approach of mConcAppt. A *clear and limited scope of functionality* is explicitly considered during the interaction design specification. *Enhancement of existing business processes, usage context and environment*, as well as *consistent look and feel* are ensured by the systematic evaluation of the as-is and to-be situations in combination with prospective end users. *Performing early usability testing* is ensured by conducting concrete user tests after each iteration cycle of the concept similar to functional testing in agile development processes.

Dealing with *limited user attention* is still not explicitly addressed by mConcAppt, but should be considered in future work. The general challenges as mentioned in Section 1 are implicitly covered by the complete mConcAppt method with the essential approach followed during the design of the method being user-centered and lightweight. Even if our practical experience with mConcAppt is mainly based on the development of mobile business apps, the circumstance of addressing general challenges enables mConcAppt to be applicable in mobile app development in general. The main requirements engineering work is performed in the context of a one-day workshop, respectively its pre- and post-processing. The workshop strongly focuses on the requirements of (lead) users of the mobile business app and elicits as-is and to-be situations, but also involves, for instance, customers, business analysts, or architects. The results of the workshop depend on its participants, its moderator, and the planning performed up-front (see up-front activities of mConcAppt description). Our experience from various applications in industry shows that by conducting the workshop according to the fixed agenda and adhering to the additional guidelines proposed in the mConcAppt method; at least 80% of the user requirements can be elicited in the one-day workshop and feedback from other involved stakeholders can be gathered directly. If required, the elicitation of the remaining requirements and the discussion of open issues can be done after the workshop, typically in short phone conferences.

Interaction design can already start in the requirements engineering workshop, for instance if examples of screens are quickly sketched to illustrate certain ideas. But the main interaction design work is performed by the interaction designer in the back-office in close collaboration with the lead users and other involved stakeholders. Feedback on micro-iterations during interaction design can be discussed with a colleague of the responsible interaction designer. This often takes only some minutes but can be extremely helpful. Lead users and other stakeholders are also often asked for feedback. One week after the requirements engineering workshop at the latest, first sketches of the interaction design should be provided to them for feedback.

Our experience has shown that the mConcAppt method supports organizations in delivering an interaction concept ready to be implemented within two to four weeks. Based on the interaction concept, development teams were able to produce a first release of the mobile business app fulfilling the required core functionality and the usability and user experience requirements in two more weeks. Our customers were always very much satisfied with the duration of four to six weeks until a first release of a new mobile business app could be delivered.

The mConcAppt method has various interfaces to other activities like visual design, architectural design, implementation, or testing. As part of our future work, such interfaces need to be defined and elaborated in more detail. The artifacts exchanged between interaction designers, architects, and other stakeholders in the development process must be specified more precisely as well as the process of collaboration, for instance when to exchange which kinds of artifacts, provide feedback, etc. The communication interface to architectural design, in particular, needs to be addressed. As integration into existing IT infrastructures was mentioned before as a major challenge, interaction designers and architects should align solution ideas early on to assure that the IT infrastructure can fulfill the requirements arising from such solution ideas.

Producing great mobile business apps requires creativity. During the requirements engineering workshop and the elaboration of the interaction concept, creativity is required to define innovative solutions. In the future, the mConcAppt method will be extended by means of specific creativity techniques that can be adopted during the requirements engineering workshop to facilitate the elaboration of innovative solutions for mobile business apps.

As mentioned in the introduction, *support of various mobile device platforms* is another challenge for mobile business apps. The mConcAppt method will be extended to allow coping with the specifics of various mobile device platforms. In the project UID4Mobile, solutions are being developed to address the scalability of interaction concepts to different mobile device platforms. Scalability means the instantiation or tailoring of interaction concepts or designs to different mobile device platforms like Apple's iOS, Google's Android, or Microsoft's Windows Phone, but also to different device types for each platform, such as smartphones and tablets.

References

1. Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J., Korkala, M., Koskela, J., Kyllönen, P., Salo, O.: Mobile-D: an agile approach for mobile application development. In: Proc. OOPSLA 2004, pp. 174–175. ACM Press (2004)
2. Adam, S., Doerr, J., Eisenbarth, M., Groß, A.: Using Task-oriented Requirements Engineering. In: Proc. RE 2009, pp. 267–272 (2009)
3. Andersson, B., Henningsson, S.: Developing Mobile Information Systems: Managing Additional Aspects. In: Proc. ECIS 2010 (2010)
4. Carroll, J.M.: Making use: Scenario-based design of human-computer interactions. The MIT Press, Cambridge (2000)
5. Clark, J.: Tapworthy: Designing Great iPhone Apps. O'Reilly Media, Inc. (2010)
6. Cooper, A.: The Inmates Are Running the Asylum: Why High-Tech Products Drive Us Crazy and How to Restore the Sanity, Indianapolis, USA (1999)

7. de Sa, M., Carrico, L.: Lessons from early stages design of mobile applications. In: Proc. MobileHCI 2008, pp. 127–136. ACM Press (2008)
8. de Sa, M., Carrico, L., Duarte, C.: Mobile Interaction Design: Techniques for Early Stage In-Situ Design. In: Asai, K. (ed.) Human Computer Interaction: New Developments. InTech (2008)
9. Ginsburg, S.: Designing the iPhone User Experience: A User-Centered Approach to Sketching and Prototyping iPhone Apps. Addison-Wesley Professional (2010)
10. Goodwin, K.: Designing for the Digital Age: How to Create Human-Centered Products and Services, 1st edn. Wiley (2009)
11. Gorlenko, L., Merrick, R.: No wires attached: Usability challenges in the connected mobile world. IBM Systems Journal 42(4), 639–651 (2009)
12. Grill, T., Biel, B., Gruhn, V.: A Pattern Approach to Mobile Interaction Design (Die Verwendung von Patterns beim Entwerfen mobiler Applikationen). IT - Information Technology 51(2), 93–101 (2009)
13. Hess, S., Kiefer, F.: Quality by Construction through mConcAppt – Towards Using UI-Construction as Driver for High Quality Mobile App Engineering. In: Proc. of QUATIC 2012 (2012)
14. Hippel, E.: Lead Users: A Source of Novel Product Concepts. Management Science 32(7), 791–806 (1986)
15. Huang, K.-Y.: Challenges in human–computer interaction design for mobile devices. In: Proc. of the World Congress on Engineering and Computer Science 2009, San Francisco, USA, October 20-22, vol. 1 (2009)
16. iOS Human Interface Guidelines. Technical report, Apple Inc. (January 2011)
17. Jacobson, I.: Use cases - Yesterday, today, and tomorrow. In: Software and Systems Modeling, pp. 210–220 (2004)
18. Kelley, J.F.: An empirical methodology for writing user-friendly natural language computer applications. In: Proc. of ACM SIG-CHI 1983 Human Factors in Computing Systems, Boston, pp. 193–196. ACM, New York (1983)
19. Looije, R., te Brake, G., Neerincx, M.: Usability engineering for mobile maps. In: Proc. of Mobility 2007, pp. 532–539. ACM, New York (2007)
20. Mayhew, D.: The Usability Engineering Lifecylce: A Practitioner's Handbook for User Interface Design, 1st edn. Interactive Technologies. Morgan Kaufmann (1999)
21. Perry, M., O'hara, K., Sellen, A., Brown, B., Harper, R.: Dealing with mobility: understanding access anytime, anywhere. ACM Transactions on Computer-Human Interaction 8(4), 323–347 (2001)
22. Yarmosh, K.: App Savvy: Turning Ideas into iPad and iPhone Apps Customers Really Want. O'Reilly Media, Inc. (2010)
23. Yuan, Y., Zheng, W.: From Stationary Work Support to Mobile Work Support: A Theoretical Framework. In: Proc. of the International Conference on Mobile Business (ICMB 2005), pp. 315–321. IEEE Computer Society, Washington, DC (2005)
24. Wassermann, A.: Software engineering issues for mobile application development. In: Proc. FoSER 2010, pp. 397–400. ACM Press (2010)

Mapping Objective-C API to Java

Arno Puder and Spoorthi D'Silva

San Francisco State University
Department of Computer Science
1600 Holloway Avenue
San Francisco, CA 94132
{arno,spoorthi}@sfsu.edu

Abstract. Apple champions the use of Objective-C for iOS development and has prohibited the use of virtual machines in the past. In previous work we have shown how Java can be used as an alternative programming language for iOS applications. We described a byte code-level cross-compiler that translates Java-based applications to portable C code to circumvent this legal restriction. A major challenge not addressed in our previous work pose the significantly growing size of the iOS API, since every Objective-C based API needs to be exposed in Java. So far we required the necessary Java skeletons to be written by hand. Since its introduction in 2007, the iOS API has nearly doubled in size. Considering the size of the iOS API this approach does not scale. In this paper we describe an API mapping tool that can generate the required Java skeletons by parsing Objective-C header files. Emphasis is placed on mapping Objective-C's dynamically typed API to strongly typed API in Java.

Keywords: API Mapping, Objective-C, Java, iOS.

1 Introduction

With the introduction of the iPhone in 2007 the smartphone market has witnessed significant growth, slowly cutting into the traditional cell phone business. Apple has created an immensely successful ecosystem around iOS with the App-Store. Apple offers an Objective-C based development environment for iOS apps. The tremendous growth of the iOS platform is reflected by the size of its API (Application Programming Interface). Table 1 gives a quantitative comparison on the number of classes and methods between different versions of iOS. It should be noted that iOS still exposes a significant amount of C API via functions and structs besides the Objective-C based API. As can be seen, the iOS API has nearly doubled in size since its first public release.

Apple has long resisted the use of programming languages other than Objective-C. Amongst others, Apple forbade the use of virtual machines on iOS. In a previous project, we designed and implemented a byte-code level cross-compiler to translate Java classes to portable C code, thus circumventing Apple's restrictions regarding virtual machines [12]. Cross-compilation is necessary but

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 21–43, 2013.

not sufficient for developing Java-based iOS applications. An application also depends on the rich iOS API that must be exposed in Java.

In the past, we have manually designed and written Java skeletons to provide a Java API for iOS. Considering the substantial size of the iOS API, the process of manually writing these skeletons does not scale. As a continuation of our previous work, we describe a API mapping tool in this paper that parses Objective-C header files to generate Java skeletons. Doing so requires to solve two major issues: firstly, mapping API based on a dynamically-typed language (Objective-C) to a statically-typed API (Java). Secondly, creating "glue code" that bridges between the different object models of the two languages at runtime.

Table 1. iOS API statistics

	iOS 3.2	**iOS 4.2**	**iOS 5.0**
Classes	282	423	503
Protocols	63	93	112
Instance Methods	4357	5756	7021
Static Methods	470	637	759
Functions	2041	2484	2714
Structs	236	289	426

This paper is organized as follows: Section 2 provides a brief overview of Objective-C and iOS. Section 3 discusses related work. In Section 4 we give a detailed overview of our API mapping tool. Section 5 provides a side-by-side comparison of a simple application written in Objective-C and the same application written in Java using our generated API. Finally, in Section 6 we give a brief conclusion and an outlook to future work.

2 Background

In this section we provide a very high-level overview of Objective-C and iOS. It is not meant as an exhaustive introduction to either topic which is out of scope for this paper. The purpose of this section is to give sufficient background information on both Objective-C and iOS to motivate the challenges mapping Objective-C API to Java. The examples given in the following highlight the corner cases that must be dealt with during the API mapping. A subsequent section will describe our solutions to the various challenges.

2.1 Objective-C

Objective-C was initially designed by Brad Cox in the early eighties. It is inspired by Smalltalk basing its object model on dynamic typing. NeXT Software licensed the Objective-C language in 1988 and developed its operating system called NEXTSTEP. In the mid-nineties, Apple acquired NeXT Software and the

language became the language of choice for application development. With the introduction of iOS in 2007 the language has seen a renaissance and has become the main development language for iOS app development. Apple offers its own integrated development environment based on Objective-C through Xcode.

Similar to C++, Objective-C is a strict superset of the C programming language adding object oriented features [7]. Any legal C program is also a legal Objective-C program. All C libraries are available in Objective-C and the entry point of an Objective-C program is defined via the usual `main()` function. The following code excerpt shows how to define a class in Objective-C:

Objective-C class declaration and definition

```
1 @interface Fraction: NSObject {
2     int numerator;
3     int denominator;
4 }
5 -(void) setNumerator:(int)n;
6 -(void) setDenominator:(int)d;
7 -(int) numerator;
8 -(int) denominator;
9 @end
10
11 @implementation Fraction
12 -(void) setNumerator:(int)n { numerator = n; }
13 -(void) setDenominator:(int)d { denominator = d; }
14 -(int) denominator { return denominator; }
15 -(int) numerator { return numerator; }
16 @end
17
18 // Using class Fraction
19 Fraction* frac = [[Fraction alloc] init];
20 [frac setNumerator:3];
21 [frac setDenominator:5];
22 printf("Numerator: %d", [frac numerator]);
```

Just like C++, Objective-C distinguishes between a *declaration* (denoted by the keyword `@interface`) and a *definition* (denoted by the keyword `@implementation`) of a class. Class `NSObject` serves as a base class, similar to `java.lang.Object`. The prefix "NS" is common for many Objective-C class names and is a relict of the origins of the class library developed by NeXT Software. Fields are declared within curly brackets followed by the methods of the class. Instance methods are prefixed with a "-" while class methods are prefixed with a "+" followed by the return type of the method. Lines 19–22 show the use of an Objective-C class. The square brackets denote a method invocation where the first argument is the target followed by the message being sent to that target. Creating a new instance requires to send the class object the `alloc` message (which only allocates sufficient memory for the instance) followed by the `init` message that serves as a constructor. Objective-C supports the notion of *named arguments*. Unlike other programming languages such as C, C++, or Java, in

Objective-C every argument is named as shown in the following addition to class `Fraction`:

Named Arguments

```
// ...
-(void) setNumerator:(int)n andDenominator:(int)d
 { numerator = n; denominator = d; }

// ...
[frac setNumerator:3 andDenominator:5];
```

Named arguments support the notion of *fluent interfaces* that make it easier for a programmer to associate semantics with a method [4]. The concatenation of the argument names `setNumerator:andDenominator:` is referred to a *selector* in Objective-C. Method dispatch is solely based on the selector. Since formal parameter types play no role in method dispatch, Objective-C does not support overloading. Mapping a selector to a Java method name would therefore be valid mapping. However, selectors in iOS tend to be lengthy, such as `tableView:willDisplayCell:forRowAtIndexPath:` from `UITableViewDelegate`. Exploiting Java's capability for method overloading leads to more poignant method names as shown in a subsequent section.

Although Objective-C is dynamically typed, it is possible to define interfaces to allow for better checks at compile time. Interfaces are defined via so-called *protocols*. While Objective-C only supports single inheritance for classes, it supports multiple inheritance for protocols. The following code excerpt shows the declaration of the protocol `Printing` that is implemented by class `Fraction`:

Objective-C protocols

```
@protocol Printing
-(void) print;
@end

@interface Fraction: NSObject <Printing>
// ...
@end

@implementation Fraction
-(void) print
 { printf("%d/%d", numerator, denominator); }
@end

// Call print method
id<Printing> obj = ...;
[obj print];
```

The protocols that a class implements are listed between < and > in the class declaration. Protocols are not first class types, i.e., they cannot be used as

type names. The declaration in line 15 above tells the Objective-C compiler that variable `obj` is a reference to an object supporting the `Printing` protocol. It is important to note that true to its dynamic nature, a class may not implement all methods declared in a protocol. Invoking an unimplemented optional method leads to a runtime error. iOS is making liberal use of optional methods. Since Java does not support optional methods in interfaces, this will be a challenge for the API mapping.

Objective-C supports the notion of class mix-ins through *categories* [13]. A category can add methods to an existing class. The name of the category is mentioned between round brackets:

Objective-C categories

```
1 @implementation Fraction (FractionCategory)
2 -(void) multiplyWith:(Fraction*)f
3 {
4     numerator *= f->numerator;
5     denominator *= f->denominator;
6 }
7 @end
```

Although categories cannot add additional fields, they can add one or more methods to an existing class, even to classes for which the source code is not available. In the example above, instances of class `Fraction` can respond to the message `multiplyWith:`. iOS makes use of categories which will need to be mapped to Java.

2.2 iOS

Since its introduction in 2007, the iOS API is largely based on Objective-C. Common to many GUI frameworks, iOS supports the Model/View/ Controller paradigm through various classes [8]. Common UI elements are defined by classes such as `UILabel` (labels), `UIButton` (buttons), or `UITextView` (input fields). Complex widgets such as `UIAlertView`, `UITableView` or `UIDatePicker` have become the trademark look-and-feel for iOS-based apps. The following excerpt from the iOS SDK shows the API to create a new `UIButton` via the static method `buttonWithType:`:

Objective-C: UIButton

```
1 typedef enum {
2     UIButtonTypeCustom = 0,
3     UIButtonTypeRoundedRect,
4     //...
5 } UIButtonType;
6
7 @interface UIButton : UIControl
8 +(id) buttonWithType:(UIButtonType)buttonType
9 //...
10 @end
```

From an API mapping perspective there are two points worth mentioning in this example. For one, even though method `buttonWithType:` returns a `UIButton` instance, the return type is declared as the generic `id` type. For a strongly typed language such as Java it would be more appropriate to map the return type to `UIButton`. Furthermore, the formal type of parameter `buttonType` is based on an enum. Even though the compiler treats the enum as an `int`, the Java API should offer identifiers for constants such as `UIButtonTypeRoundedRect`.

iOS makes heavy use of Objective-C protocols, especially to define the interface of delegates that serve as callbacks. Much of the behavior of UI widgets is driven by delegates that define different aspects of a widget. E.g., protocol `UITableViewDelegate` features 18 callback methods that control the look-and-feel of a `UITableView`. All but one of these callbacks are optional and need not be implemented. Mapping an Objective-C protocol to a Java interface would therefore incur an inconvenience to a Java developer as all 18 methods would need to be implemented.

iOS offers several classes to support various data structures. Class `NSArray` manages an array while `NSDictionary` maintains a set of key-value pairs. These classes are used by iOS GUI classes. E.g., the `UIView` class that represents the base class of the view hierarchy has a method `getSubviews` that returns a `NSArray` instance with a list of all child views. When mapping this method to a Java API the question arises if the Java counterpart should also offer Java versions of classes `NSArray` and `NSDictionary` or use the more familiar Java interfaces `java.util.List` and `java.util.Map`.

As already mentioned in the introduction, a significant portion of the iOS API is not defined through Objective-C classes but C functions. In particular the low-level graphic drawing functions of the Quartz framework are exclusively defined via C functions. As an example, consider the following iOS API part of the core graphics framework that defines a rectangular region on the screen:

C: CGRect

```
1 struct CGRect {
2     CGPoint origin;
3     CGSize  size;
4 };
5 typedef struct CGRect CGRect;
6
7 CGRect CGRectIntersection (
8     CGRect r1,
9     CGRect r2
10 );
```

A `struct` in Objective-C retains its original semantics of the C programming language and can be viewed as a value type. A set of C functions perform various operations on `CGRect` instances such as computing the intersection of two rectangles. As Java does not support functions or value types, special mapping rules must be devised to yield a natural Java API. The iOS API also makes use of features typically found in C programs. The following code excerpt shows

the use of a pointer-to-a-pointer argument to mimic call-by-reference which also needs to be handled specially in Java:

Call-by-reference

```
// Objective-C
@interface AVAudioSession : NSObject
-(BOOL) setActive:(BOOL)beActive
            error:(NSError**)outError;
//...
@end
```

Objective-C categories are used in iOS to define so-called *additions* that add features to existing classes. E.g., iOS's UIKit adds several methods to class `NSString` that are specific to UI programming. As shown below, the `UIKitAddition` category adds a method to `NSString` to determine the width and height in pixels of a string given a particular font:

UIKit addition to NSString

```
@interface NSString (UIKitAddition)
-(CGSize) sizeWithFont:(UIFont*)font;
//...
@end
```

Finally we want to mention the memory management model of iOS applications that has no impact on the generated Java API, but has significant consequences of the underlying JNI implementation. Apple has opted not to include a garbage collector in iOS. Instead, memory management is handled via reference counting [6]. Base class `NSObject` offers methods `retain` and `release` to increase and decrease the reference count of an object respectively:

Reference counting

```
Fraction* frac = [[Fraction alloc] init];
printf("Retain count: %d", [frac retainCount]);
[frac retain];  // 2
[frac release]; // 1
[frac release]; // 0. Object will be deleted
```

Proper memory management is one of the most difficult aspects of iOS programming. While iOS 5 has introduced ARC (Automatic Reference Counting) where the compiler automatically inserts `retain`/`release` methods, this feature is not supported by the core framework and it will also not work for multithreaded applications. Making Java's garbage collector work in conjunction with reference counting is a major challenge of the cross-compiled Java application.

In this section we have highlighted various features of the Objective-C programming language and the iOS API that have impact on the design of a corresponding Java API. After discussing some related work, we will describe our

mapping tool that can generate a Java API taking C/Objective-C features such as structs, categories or named parameters into account.

3 Related Work

API mapping from one language to another is not new. There are several ongoing efforts in this area. In this section we will discuss some prominent examples, some of which are not in the mobile space. One example is SWIG (Simplified Wrapper and Interface Generator) that is used to bind the code written in C/C++ to a variety of languages like Tcl, Perl, Java, C#, etc [3,1]. SWIG does this by automatically generating the code required for the binding using a simple interface file as input. It tries to provide a complete mapping for the APIs exposed in this interface file that needs to be provided by a developer. Since SWIG does not have a memory manager of its own, special care needs to be taken in SWIG to avoid releasing the associated C++ objects that still might be used. SWIG provides directives to handle special cases but does not make use of any heuristics for mapping the API.

Similar to SWIG, CORBA (Common Object Request Broker Architecture) helps with mapping API of remote objects in a distributed environment [11]. CORBA makes use of a specialized language called IDL (Interface Definition Language) that can be used to define the interface of remote objects. IDL is not tied to a specific programming language and the CORBA specification defines mapping rules from IDL to various high-level languages. An IDL compiler creates stubs and skeletons from a specific IDL file that are used to carry out the client server communication. Stubs are generated in the same language as the client while the skeletons are generated in the same language as the server. Client and server do not need to be implemented in the same language.

While the tools discussed so far do not directly relate to mobile applications, there are tools such as MonoTouch that allow developers to create iOS applications using C# and .NET [14]. MonoTouch uses a so-called API Definition file that consists of C# interfaces with special annotations and attributes for the binding purpose. These special annotations handle the exceptional cases for the API mapping such as static methods, binding rules for properties and so on. MonoTouch has its own garbage collector and also provides users with options to release resources explicitly before the garbage collectors collects them.

While MonoTouch concentrates on offering a binding between the C# and Objective-C, there are tools that try to provide cross-platform support. One such tool is PhoneGap that supports JavaScript API that acts as the bridge to the native platform [2]. PhoneGap provides functionality to develop more than just a web app in the sense that the JavaScript APIs also provide access to the phone's hardware such as accelerometer, camera, etc.

PhoneGap provides a generic interface across multiple platforms using HTML, JavaScript and CSS. By default, the API support is only for certain hardware functions and and does not cover a 1:1 mapping of the various mobile platform APIs. One thing to note is that PhoneGap does support a wide range of

mobile platforms like iOS, Android, Blackberry to name a few. In order to perform complex tasks, a developer can write custom plugins by writing custom JavaScript APIs and custom native components so that JavaScript invocations can be delegated to the native code.

While PhoneGap and MonoTouch provide native API access in a different programming language, JamVM, a light-weight implementation of the Java Virtual Machine enables developers to run Java programs on iOS [10]. JamVM for iOS uses reflection in Objective-C to extract API information from the binary iOS libraries. Based on that information, JamVM for iOS generates JNI code that a Java application needs to make calls to iOS. Since JamVM for iOS effectively parses API from a binary library, it is impossible to extract symbolic information such as type identifiers resulting in unnatural looking Java API. Furthermore, JamVM for iOS is not capable of generating Java API from C functions.

Unlike some of the tools mentioned above that only map a subset of the native APIs, our tool tries to provide complete mapping for all the APIs that are available in iOS. Our tool automates the process of generating the API mappings. It not only handles exceptions by providing special hints but also tries to create more natural looking APIs to support Java-ism by using special heuristics. The application developer need not take special care for releasing the resources while using our tool since our API mapping tool provides its own memory management mechanism.

Table 2 summarizes the features of the various tools discussed above and compares them to our API mapping tool.

Table 2. Comparison of related work

	SWIG	CORBA IDL	MonoTouch	PhoneGap	JamVM for iOS	Our API mapping tool
Input Artifact	Header	IDL	Header	n.a	Binary Library	Header
API mapping mechanism	Automated	Automated	Automated	Manual	Automated	Automated
Language Association	1:n	m:n	1:1	n:1	1:1	1:1
Memory Management	Manual	Manual	Automatic	Automatic	Automatic	Automatic
Handling Anomalies	Directives	n.a	Annotations	n.a	Not Present	Advisor
Scope	Complete	Complete	Complete	Subset	Subset	Complete
Heuristics for natural looking API	✗	✗	✗	n.a	✗	✔

4 API Mapping

This section introduces our API mapping tool. Section 4.1 will first discuss the design goals and the architecture of our tool. Section 4.2 describes the Java API that our tool generates. The Java API needs to be complemented by matching JNI code that acts as a bridge between the Java application and iOS.

The generation of the JNI code is described in Section 4.3. In Section 4.4 we give a brief overview of the external advise needed by our tool.

4.1 Overview

Before giving an in-depth overview of our API mapping tool, we first need to establish the goals of its design. Ultimately the purpose of the tool is to avoid having to manually implement the Java API whenever Apple releases a new version of iOS. At the same time, Java and Objective-C are quite different languages in the way they implement their respective object model. When deriving a Java API from its Objective-C original, one should take those differences into account. Specifically, our mapping tool strives to fulfill the following design goals:

1. Automate API mapping.
2. Map all relevant iOS API.
3. Keep naming conventions.
4. Support Java-isms.

First and foremost the API mapping should be automated. Given the fact that iOS evolves rapidly from version to version manually mapping new Objective-C API to Java does not scale. Our second goal is to map all relevant iOS API to increase coverage. This implies that we not only map Objective-C API but also the significant number of C functions part of the iOS API. Our third goal is to make the transition from Objective-C to Java as seamless as possible. By keeping naming conventions it will be easier to transition to Java for iOS development as the original documentation from Apple can serve as a reference even for the Java API.

The most important goal is to support conventions that Java developers have become accustomed to. For one, the API should make use of strongly typed interfaces where appropriate. Due to the nature of Objective-C's dynamic typing, the API is often less specific compared to an equivalent Java API. Furthermore, we try to make use of familiar J2SE API instead of iOS specific data structures. Last but not least, Java developers expect the presence of a garbage collector that will need to interface with the reference counting mechanism used by iOS.

Figure 1 gives an overview of the architecture of our API mapping tool. Common to any compiler, the mapping tool can be divided into a frontend and a backend. The frontend reads and parses the original iOS SDK header files to scan it for relevant API. As mentioned earlier, not all required information can be derived from those header files (such as the specific types of arguments). In these cases, external *advise* needs to be provided to the mapping tool. Based on the header files and the advise, the backend generates two kinds of artifacts: the Java API that can be used by an iOS app developer and the JNI code that acts as a bridge between the Java application and iOS. All aspects of the API mapping tool are discussed in detail in the following sections.

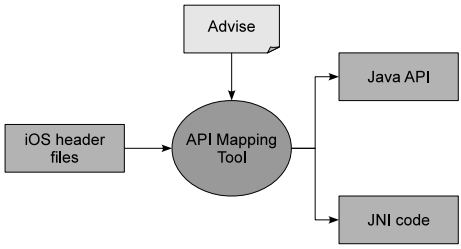


Fig. 1. API Mapping Tool

4.2 Generating Java API

In the following we focus on how a Java API can be derived from the original C/Objective-C API. As C is the foundation of Objective-C, we first discuss how to map certain features of C to Java followed by a description of the mapping of Objective-C API.

Mapping of C API. Primitive types are mapped to their natural counterparts. It is common in C to typedef names for commonly used primitive types since their size is not defined by the C standard (e.g., `sizeof(int)` depends on the underlying architecture). iOS has typedefs for various primitive types such as `BOOL`, `int8_t`, `int16_t`, and `int32_t` that are mapped to their Java counterparts `boolean`, `byte`, `short`, and `int`. Signed and unsigned integer values have to be treated the same as Java does not offer support for unsigned values. Defining a platform-independent type such as `int32_t` is usually done by `#ifdef`s that check for different architectures. In order to avoid parsing preprocessor directives, we provide this knowledge via external advise.

Many iOS data structures are defined via C structs such as `CGRect` introduced in Section 2.2. A `CGRect` defines a 2-dimensional region in the iOS core graphics framework. Its members are based on two other structs, `CGPoint` and `CGSize`, for the position and the size of the region. Since Java does not offer support for value types, structs are mapped to Java classes, making the members of the struct public fields of these classes.

The iOS API uses very regular and dependable naming conventions. In Section 2.2 we mentioned function `CGRectIntersection` that computes the intersection of two `CGRect` instances. The name of all C functions that operate on a particular struct are always prefixed with the name of that struct in iOS. The first argument is an instance of that struct. We take advantage of these regular naming conventions by making these C functions the methods in the generated Java class. The first parameter becomes the implicit `this`-parameter. The following code excerpt shows the generated Java version of `CGRect`:

Java version of CGRect

```
1 public class CGRect {
2     public CGPoint origin;
3     public CGSize  size;
4
5     native public CGRect intersection(CGRect r2);
6     //...
7 }
```

We take advantage of similar naming conventions when mapping enums. The common prefix of enum members is removed since Java enums define a scope in contrast to C enums. The following code shows the mapping of `UIButtonType` introduced in Section 2.2:

Mapping of an enum

```
1 // Objective-C
2 typedef enum {
3     UIButtonTypeCustom = 0,
4     UIButtonTypeRoundedRect,
5     //...
6 } UIButtonType;
7
8 // Java
9 public enum UIButtonType {
10     Custom(0),
11     RoundedRect,
12     //...
13 };
```

It should be noted that the rules to take advantage of naming conventions are hard-coded heuristics and not part of the advise used by our tool. If an API does not make use of those naming conventions, functions are mapped to static methods of a global class.

Mapping of Objective-C API. By default, an Objective-C class is mapped to a Java class of the same name. A selector in Objective-C is mapped to a Java method. While it would be possible to use the selector as the name of a Java method, it would result in unnatural names. Since Java does not support named arguments, using the selector as the name for a method could be confusing. We use various heuristics to generate more Java-like method names taking advantage of method overloading in Java. By default, the first component of a selector is used as the name of the method. As an example, consider the following methods of class `NSString`:

Mapping of NSString

```
1 // Objective-C
2 - (int)compare:(NSString *)string;
3 - (int)compare:(NSString *)string
```

```
4         options:(NSStringCompareOptions)mask;
5 - (int)compare:(NSString *)string
6         options:(NSStringCompareOptions)mask
7           range:(NSRange)compareRange;
8 - (int)compare:(NSString *)string
9         options:(NSStringCompareOptions)mask
10          range:(NSRange)compareRange
11         locale:(id)locale;
12
13 // Generated Java methods
14 public int compare(NSString string);
15 public int compare(NSString string, int mask);
16 public int compare(NSString string, int mask,
17                    NSRange compareRange);
18 public int compare(String string, int mask,
19                    NSRange compareRange,
20                    Object locale);
```

Class `NSString` features four different methods to compare two strings. Instead of using the selector as a method name only the first component of the selector is used. For this reason selector `compare:options:` is mapped to method name `compare` in Java. This is possible because the formal parameter types are different for all four selectors and one can rely on Java's method overloading to distinguish among them. If the formal parameter types are identical for different selectors, our heuristic will append the second component of the selector to the generated method name until the Java methods are distinguishable again.

Parameter types are mapped to their counterpart. The Objective-C header parser resolves typedefs to determine the underlying type. Since Java does not support typedefs, the original type is used when generating the Java API. This can be seen in the example above where `NSStringCompareOptions` is a typedef for an `int`.

Various other heuristics determine the mapping for formal parameter types. E.g., the generic Objective-C type `id` is mapped to `java.lang.Object`. For parameters whose type is a pointer-to-a-pointer we assume that it is an output argument. In this case a so-called *holder class* is generated that acts as a wrapper for the actual output value. We make use of Java generics to create a type-safe parameter type. As an example, `NSError**` in the Objective-C API would be mapped to `Reference<NSError>` on the Java side.

As expressed in the design goals for the API mapping, we strive towards natural looking Java APIs. For this reason Objective-C data structures are mapped to their J2SE counterpart. E.g., `NSArray` will be mapped to `java.util.List` and `NSDictionary` will be mapped to `java.util.Map`. There are three main reasons for doing so:

Usability: Java programmers know how to deal with Java data structures. They know the API and are good at using it. Using the original Objective-C

requires to learn new API. Furthermore, making use of J2SE API has the benefit of being able to use Java-isms such as for-each loops.

Performance: Cross-compiling a Java implementation for `java.util.List` to C code is more efficient than having a wrapper for `NSArray` where each method invocation on the Java side has to be forwarded to an appropriate method of `NSArray`. The one extra level of indirection incurs avoidable overhead.

Scope: The question arises where to draw the line. E.g., using a Java version of `NSString` instead of `java.lang.String` would result in unnatural Java programs, especially since Java string literals are of type `java.lang.String`.

For those reasons we have opted to map Objective-C classes to their natural J2SE counterpart. Java versions of classes such as `NSArray` and `NSString` are available, however, if used as formal argument types in method signatures they are mapped to the equivalent J2SE type. Mapping of generic parameter types such as `NSArray` and `NSDictionary` leads to another challenge. Due to Objective-C's support for dynamic typing those data structures can hold objects of arbitrary type. This would be equivalent to `List<Object>` and `Map<Object, Object>` in Java. However, in many cases the type of the objects held in a container is restricted. E.g., the `getSubviews` method of class `UIView` returns a list of children of a particular `UIView` instance. As per iOS documentation each child must be an instance of a subclass of `UIView`. Objective-C is not capable of expressing this restriction in a method's signature. However, in Java it would be possible to restrict the return type of `getSubviews` by making use of generics to `List<UIView>`. We have opted to make use of Java generics and support the mapping of restricted container types. Since this information is not discernable from the Objective-C header files of iOS, it must be provided externally as an advise to the API mapping tool.

Categories were introduced as a mechanism in Objective-C to add methods to existing classes and as was shown in Section 2.2, iOS makes use of this in the UIKit Additions. Since Java does not support mix-in classes, the way we handle additions is by inlining the methods to the class to which they are added via a category. Another challenge is the mapping of Objective-C protocols. By default, an Objective-C protocol is mapped to a Java interface. This requires an app developer to implement all methods of the interface. Since many methods are often optional in iOS, we also emit an *adapter class* in Java that implements the interface [5]. Optional methods are given a default implementation in the adapter class and only mandatory methods are left abstract.

4.3 Generating Java JNI

The previous section discussed the mapping of Objective-C API to Java. The result is a natural looking Java API for iOS that can be used by an app developer. Methods of the Java skeleton classes are labeled as `native` and running a Java-based iOS app requires JNI code that acts as a bridge between the Java application and iOS [9]. Our mapping tool also generates the necessary JNI code automatically. The details are discussed in the following.

Wrapping. When a Java application uses a class from the iOS library, it inadvertently needs to use a native Objective-C object. As an example, consider class `UIAlertView`. In iOS, this class is used to pop open an alert view as a modal dialog, often presenting the user with several options. Instantiating the Java class `UIAlertView` must lead to an instantiation of the Objective-C class `UIAlertView`. Subsequent method invocations on the Java object need to be forwarded to its Objective-C counterpart. The following code shows the effect of executing Java code in relationship to what needs to happen on the native layer:

Effects of Java method invocations

```
1 // Java (Configuration of UIAlertView not shown here)
2 UIAlertView alert = new UIAlertView();
3 alert.show();
4
5 // Objective-C
6 UIAlertView* alert = [[UIAlertView alloc] init];
7 [alert show];
```

The Java object can be seen as a wrapper or proxy for the underlying native Objective-C object. Method invocations on the Java object are forwarded to the wrapped Objective-C object. This happens transparently to the app developer. Figure 2 illustrates the notion of a wrapper. White circles represent pure Java objects. Those objects are part of the application and have no relationship with iOS. Objective-C objects are represented by black dots. Some Objective-C objects exist only inside iOS and are not visible to the application. Other Objective-C objects such as the `UIAlertView` instance are exposed to the Java application via a wrapper.

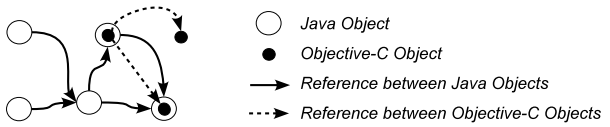


Fig. 2. Wrapping native Objective-C objects

It should be obvious that there needs to be a tight association between a wrapper object and the wrapped native Objective-C object. This association is bijective and it must be possible to access the native object from the wrapper and vice versa. We make use of JNI to forward an invocation to a Java method to the underlying Objective-C object. All methods in the generated Java API are declared as native to allow the injection of Objective-C code. The following code excerpt shows the native implementation of the two methods of `UIAlertView` used in the example earlier:

JNI implementation

```
1 void UIAlertView___INIT___(JAVA_OBJECT me)
2 {
```

```
3     UIAlertView* obj = [[UIAlertView alloc] init];
4     ASSOCIATE(me, obj);
5 }
6
7 void UIAlertView_show__(JAVA_OBJECT me)
8 {
9     UIAlertView* thiz = GET_ASSOCIATED_NATIVE(me);
10    [thiz show];
11 }
```

The JNI code shown here deviates from the official JNI specification in order to simplify the example. Note that JNI requires C functions for native Java methods, however, it is possible to make use of Objective-C for the implementation of those functions. Instantiating class `UIAlertView` yields in the invocation of `UIAlertView__INIT__` via JNI where the corresponding Objective-C version of `UIAlertView` is created via the usual `alloc`/`init` messages. Function `ASSOCIATE` is part of our runtime library and its purpose is to create the aforementioned bijective link between the Java object and the wrapped Objective-C instance. When the Java application invokes a method on the `UIAlertView` instance later, helper function `GET_ASSOCIATED_NATIVE` is used to retrieve the associated Objective-C object. It should be emphasized again that the JNI code shown above was also generated by our mapping tool.

Upcalls. The previous section dealt with the case where the application is making a downcall to iOS. In some instances the opposite happens: iOS makes an upcall to the application. This occurs frequently when iOS delivers events such as touch, timer, or sensor events. E.g., the `UIAlertView` calls the application whenever the user tapped on a button. In order to receive events, the application needs to register an appropriate callback, commonly referred to as a delegate. iOS makes use of Objective-C protocols for this purpose. Using the `UIAlertView` as an example again, iOS defines the protocol `UIAlertViewDelegate` that allows an application to respond to button-tap events.

What makes upcalls different from downcalls is the fact that for protocols the application needs to provide an implementation, not iOS. Mapping an Objective-C protocol to a Java interface lets the developer implement the interface, however, the resulting implementation only exists in Java space. As iOS has no knowledge of Java, a different kind of wrapper is needed. In the previous section we introduced a Java wrapper for an Objective-C object. Now it is just the reverse: what is needed is an Objective-C wrapper for a Java object. Figure 3 illustrates the reversal of roles.

From the perspective of iOS, a delegate needs to be an Objective-C object. The purpose of the delegate wrapper is to forward calls made by iOS to the Java application. The mapping tool can automatically generate these delegate wrapper for each Objective-C protocol based on the protocol's declarations. The following code excerpt shows the code generated for the declaration of `UIAlertViewDelegate`:

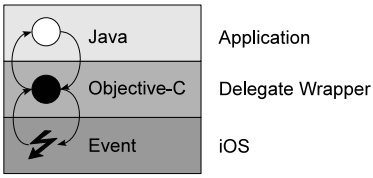


Fig. 3. Delegates in iOS

Objective-C wrapper for UIAlertViewDelegate

```
@interface UIAlertViewDelegateWrapper :
                      NSObject <UIAlertViewDelegate> {
    JAVA_OBJECT* delegate;
}

      -(void)alertView:(UIAlertView *)alertView
  clickedButtonAtIndex:(NSInteger)buttonIndex;
//...
@end
```

Class `UIAlertViewDelegateWrapper` is created by the mapping tool and conforms to the `UIAlertViewDelegate` protocol. The field `delegate` is a reference to the Java implementation of `UIAlertViewDelegate`. When the Java application sets a delegate, the above Objective-C class will be instantiated and registered with iOS. iOS will send the message `alertView: clickedButtonAtIndex:` whenever the user taps on a button. Not shown here is the implementation of `UIAlertViewDelegateWrapper` that is also created by the mapping tool. The implementation uses JNI to make the upcall to the corresponding Java method with `delegate` as the target.

A related problem occurs with upcalls made to regular iOS classes. E.g., class `UIView` offers a method called `drawRect:`. This method will be called by iOS whenever the surface of the `UIView` needs to be redrawn. An application can subclass `UIView` and override `drawRect:` to redraw itself. Here the situation is related to protocols in the sense that the subclass will be implemented in Java by the developer. As iOS has no knowledge of Java, a wrapper needs to intercept calls made by iOS to `drawRect:` and forward it to the corresponding Java implementation. The following code fragment shows the wrapper generated by our API mapping tool for class `UIView`:

Class Callbacks

```
// Objective-C wrapper
@interface UIViewWrapper : UIView
-(void) drawRect:(CGRect)rect;
@end

// JNI constructor for UIView
void UIView___INIT___(JAVA_OBJECT me)
```

```
8 {
9     UIView* obj = [[UIViewWrapper alloc] init];
10    ASSOCIATE(me, obj);
11 }
```

Class `UIViewWrapper` is derived from `UIView` to allow it to override method `drawRect:`. Not shown here is the implementation of `UIViewWrapper` that forwards calls to the corresponding Java implementation of method `drawRect()` via JNI. When a Java application instantiates a `UIView`, the JNI implementation of the `UIView` constructor will instantiate a `UIViewWrapper` instead. Wrapper classes will need to be generated by our mapping tool whenever a class features a callback such as `drawRect:`. This knowledge cannot be derived from the Objective-C header files and consequently needs to be provided via an external advise.

Memory Management. Earlier we mentioned the bijective relationship between the wrapper object and the native Objective-C object. In the following we focus on memory management issues as iOS and Java are governed by different memory management mechanisms. As explained earlier, iOS uses reference counting while Java makes use of a garbage collector. In terms of object lifecycle, a wrapper object cannot exist without the Objective-C object it wraps. Assume the example from the previous section. The application instantiates class `UIAlertView` that triggers the creation of an Objective-C object. iOS expects the application to release the Objective-C object in order to signal that it no longer needs the `UIAlertView` instance. This can be accomplished by registering a finalizer for the wrapper object. In Java, overriding method `finalize()` will cause the garbage collector to invoke this method just before the object will get collected. Making this method native allows again to inject code on the native level via JNI:

Finalization

```
1 // Generated Java
2 class UIAlertView {
3     //...
4     native protected void finalize();
5 };
6
7 // Generated JNI
8 void UIAlertView_finalize__(JAVA_OBJECT me)
9 {
10    UIAlertView* thiz = GET_ASSOCIATED_NATIVE(me);
11    [thiz release];
12 }
```

Function `UIAlertView_finalize__` will be called by the garbage collector just before the wrapper object is being collected. The associated native Objective-C object is retrieved via `GET_ASSOCIATED_NATIVE` followed by sending it the

`release` message. Note that the latter will not necessarily destroy the Objective-C object. If iOS itself holds a reference to the same object the reference count would not drop to zero therefore keeping the object alive. Whenever iOS no longer needs the object, it will send it the `release` message as well eventually destroying it. Therefore, an Objective-C object can outlive its wrapper but not vice versa.

So far we have discussed the relationship between a wrapper and the wrapped native Objective-C object. However, in terms of memory management one needs to consider relationship between wrapper objects as well. The unfortunate fact is that iOS will not always retain objects where one would expect. As an example, consider the protocol `UIAlertViewDelegate` that defines the API for a delegate used by `UIAlertView`. The `UIAlertView` allows the setting of a `UIAlertViewDelegate` instance that acts as a callback that iOS will invoke once the user has tapped on a button. Even though the `UIAlertViewDelegate` is set via a setter on the `UIAlertView`, the latter will not retain the delegate. This is the applications responsibility in iOS. Without arguing the merit of such an API design, the problem is that the following Java code using the Java-based iOS API would not work:

Setting a UIAlertViewDelegate

```
1 UIAlertView alert = //...
2 alert.setDelegate(new UIAlertViewDelegate() {
3     // React to button click of the UIAlertView
4 });
```

In Java it is common to use anonymous classes in these cases as shown in the code excerpt above. A Java programmer would expect that the delegate is saved internally. However, since `UIAlertView` is merely a wrapper that passes the invocation of `setDelegate` to the native Objective-C object, this will not happen in this particular case. The consequence is that the garbage collector will eventually reclaim the Java wrapper for `UIAlertViewDelegate` whose finalizer will release the underlying Objective-C instance. Since the `UIAlertView` did not retain the delegate, it will effectively be deleted even though it might still be needed. The result will be a segmentation fault since an invocation will be made on an object that does not exist anymore.

Parsing the Objective-C API one cannot identify such cases. The fact that an `UIAlertView` does not retain its delegate can only be known by studying the documentation. For the purpose of the API mapping, this knowledge needs to be passed to the mapping tool via an external advise.

4.4 Handling Exceptions

Although our API mapping tool tries to derive all necessary information from the Objective-C header files, in some cases this is not sufficient in order to generate the Java API and JNI code. Throughout the previous sections we gave several examples. The knowledge of such details is only contained in the documentation

that cannot be parsed by an automated tool. For this reason we have introduced the notion of *advise* that can be given to the mapping tool (see Figure 1 in Section 4.1). Someone familiar with the iOS API has to update the advise whenever a new version of iOS is released.

Advise can be provided to different components of our API mapping tool: the frontend that parses the header files as well as the two backends that generate code for the Java API and the necessary JNI code. During each stage of the translation process the relevant sections of the advise is consulted. The format of the advise is XML that complies to a particular schema. The following excerpt shows some of the information contained in the advise:

Advise

```
<replace pattern="//.*"/>
<replace pattern="__BEGIN_DECLS"/>
<replace pattern="__END_DECLS"/>
<!-- ... -->

<typedef java="boolean" c="BOOL"/>
<typedef java="byte" c="int8_t"/>
<typedef java="List" c="NSArray"/>
<typedef java="Map" c="NSDictionary"/>
<!-- ... -->

<class name="UIApplicationDelegate">
  <selector
   name="-application:didFinishLaunchingWithOptions:">
     <arg position="1" type="Map<String,String>"/>
  </selector>
</class>

<class name="UIView">
  <selector name="-drawRect:" delegate="true"/>
</class>

<class name="UIAlertView">
  <selector name="-setDelegate:" retain="true"/>
</class>

<class name="NSString">
  <injected-method name="toString" modifier="public">
    <signature>
      <return type="String"/>
    </signature>
    <code language="c">
      <![CDATA[
        NSString* thiz = GET_ASSOCIATED_NATIVE(me);
        return CONVERT_TO_STRING(thiz);
      ]]>
    </code>
```

```
38   </injected-method>
39 </class>
40 <!-- ... -->
```

The <replace> tags help the frontend to clean up the Objective-C header files before parsing further information (lines 1-3). These tags define regular expressions such as comment markers (line 1) that will be removed. The <typedef> tag defines basic type mapping rules. It can be used to define mappings for primitive types (lines 6 and 7) as well as class types (lines 8 and 9).

Advise that relates to a particular class is grouped with the help of the <class> tag. The methods for which advise is given are identified by the <selector> tag that references the usual Objective-C selector (lines 13, 20, and 24). Various XML attributes provide specific information about the method that cannot be derived by parsing the Objective-C header files. Attribute `delegate` (line 20) will trigger the generation of a delegate wrapper as explained in Section 4.3. Attribute `retain` (line 24) will make sure that a Java reference is held to the argument to prevent the garbage collector from reclaiming it as explained in Section 4.3.

The advise given for `UIApplicationDelegate` specifies that the type of the second argument can be narrowed to `Map<String, String>` (line 15). An example for code injection is shown for class `NSString` (line 27). This class is the iOS counterpart to `java.lang.String`. In order to easily convert a `NSString` instance to a `String`, the advise injects an additional Java method `toString()`. As can be seen in the advise above, besides the Java signature of the injected method the advise also provides the necessary JNI implementation. The latter is necessary since the knowledge of how to convert a `NSString` instance is out of scope for the mapping tool.

5 Example

In this section we make a side-by-side comparison of an iOS version of "Hello World" implemented in Objective-C and in Java using the API generated by our mapping tool. Xcode offers a tool called InterfaceBuilder that allows to define a UI using drag-and-drop through a graphical tool. InterfaceBuilder can also be used to create a boiler-plate iOS application with one click. The following code shows a programmatic version of "Hello World" to illustrate the API mapping:

iOS Hello World (Objective-C)

```
1 @implementation HelloWorldAppDelegate
2
3 -(BOOL) application:(UIApplication*)application
4    didFinishLaunchingWithOptions:(NSDictionary*)opts
5 {
6   CGRect r = [[UIScreen mainScreen] applicationFrame];
7   UIWindow* window =
8               [[UIWindow alloc] initWithFrame:r];
```

```
9   [window setBackgroundColor: [UIColor whiteColor]];
10  r.origin.x = r.origin.y = 0;
11  UILabel* label = [[UILabel alloc] initWithFrame:r];
12  [label setText:@"Hello World"];
13  [label setTextAlignment:UITextAlignmentCenter];
14  [window addSubview:label];
15  [window makeKeyAndVisible];
16  return YES;
17 }
18
19 @end
```

Below is the same application now implemented in Java using the Java API for iOS generated by our mapping tool:

iOS Hello World (Java)

```
1 public class HelloWorld extends UIApplicationDelegate {
2
3   public boolean didFinishLaunchingWithOptions(
4           UIApplication app, Map<String,String> opts) {
5     CGRect r =
6           UIScreen.mainScreen().getApplicationFrame();
7     UIWindow window = new UIWindow(r);
8     window.setBackgroundColor(UIColor.whiteColor());
9     r.origin.x = r.origin.y = 0;
10    UILabel label = new UILabel(r);
11    label.setText("Hello World");
12    label.setTextAlignment(UITextAlignment.Center);
13    window.addSubview(label);
14    window.makeKeyAndVisible();
15    return true;
16  }
17 }
```

As can be seen by direct comparison of the Objective-C and the Java version of the application, someone knowledgeable with the iOS API will immediately understand the Java implementation. Besides retaining naming conventions from the iOS Objective-C-based API, the Java version also demonstrates several mapping challenges discussed earlier: mapping of structs (`CGRect`), protocols (`UIApplicationDelegate`), and mapping of data structures to their strongly-typed J2SE counterpart (`Map<String,String>`).

6 Conclusion and Outlook

Apple favors the use of Objective-C for developing iOS applications. The purpose of our work is to give developers more freedom by offering Java as an alternative programming language. This paper focuses on the API mapping that is necessary

to expose the native Objective-C-based API in Java. Because of the sheer size of the iOS API our goal is to automate this process as much as possible. The ideas introduced in this paper have been implemented and released under an Open Source license. iOS app developers have successfully used our tool to develop Java-based apps and publish them on the Apple AppStore.

In the future we plan to focus more attention on the heuristics that lead to natural looking Java API. The heuristics currently used in our tool are based on the way Apple has designed its API (e.g., the fact that function names that operate on certain structs are prefixed with the name of that struct). We plan to investigate API outside the iOS ecosystem to derive a more general set of heuristics. Ultimately our tool should be able to create Java API for other third-party libraries.

Acknowledgements. We are greatly indebted to Paul Poley and Panayotis Katsaloulis for their invaluable insights and support.

References

1. SWIG - Simplified Wrapper and Interface Generator, `http://www.swig.org`
2. Adobe Systems. PhoneGap, `http://wiki.phonegap.com`
3. Beazley, D.: SWIG: An easy to use tool for integrating scripting languages with C and C++. In: Proceedings of the 4th Conference on USENIX Tcl/Tk Workshop, vol. 4, p. 15. USENIX Association, Berkeley (1996)
4. Fowler, M.: Domain-Specific Languages. Addison-Wesley Professional (October 2010)
5. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional (October 1994)
6. Hudak, P.: A semantic model of reference counting and its abstraction. In: Proceedings of the 1986 ACM Conference on LISP and Functional Programming, pp. 351–363. ACM, New York (1986)
7. Kochan, S.: Programming in Objective-C, 4th edn. Addison-Wesley Professional (December 2011)
8. Krasner, G., Pope, S.: A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System (1988)
9. Lindholm, T., Yellin, F.: The Java Virtual Machine Specification, 2nd edn. Addison-Wesley Pub. Co. (April 1999)
10. Lougher, R.: JamVM, `http://jamvm.sourceforge.net/`
11. Object Management Group. Common Object Request Broker Architecture, CORBA/IIOP (2004), `http://www.omg.org/technology/documents`
12. Puder, A.: Running Android Applications without a Virtual Machine. In: Venkatasubramanian, N., Getov, V., Steglich, S. (eds.) Mobilware 2011. LNICST, vol. 93, pp. 121–134. Springer, Heidelberg (2012)
13. Smaragdakis, Y., Batory, D.: Implementing Layered Designs with Mixin Layers. In: Jul, E. (ed.) ECOOP 1998. LNCS, vol. 1445, pp. 550–570. Springer, Heidelberg (1998)
14. Xamarin. MonoTouch, `http://docs.xamarin.com/ios`

A Reference Architecture for Group-Context-Aware Mobile Applications

Grace Lewis, Marc Novakouski, and Enrique Sánchez

CMU Software Engineering Institute
4500 Fifth Ave. Pittsburgh, PA 15213 USA
{glewis,novakom,eysanchez}@sei.cmu.edu

Abstract. Handheld mobile technology is reaching first responders and soldiers in the field to help with mission execution. A characteristic of mission execution environments is that people are typically deployed in teams or groups to execute the mission. Most commercially-available context-aware mobile applications are based on context expressed mainly as location and time of an individual device plus the device user's preferences or history. This work extends context to consider the group that the individual is a part of and presents a reference architecture for group-context-aware mobile applications that integrates contextual information from individuals and nearby team members operating to execute a mission. The architecture is highly extensible to support changes in context data models, context data storage mechanisms, context reasoning engines and rules, sensors, communication mechanisms and context views. A prototype implementation was built to demonstrate the validity and extensibility of the reference architecture.

Keywords: context-awareness, mobile applications, Android, software architecture, reference architecture.

1 Introduction and Motivation

Handheld mobile technology is reaching first responders and soldiers in the field to help with mission execution. These individuals operate at the tactical edge, which is a term used to describe hostile environments with limited resources, from disaster relief areas in countries like Haiti and Japan, to war zones in Afghanistan.

A major challenge at the tactical edge is getting relevant information at the time it is needed. Causes include reliance on easily misplaced paper reports, one-way information flow (up the chain of command but not down), and the lack of network bandwidth and handheld devices to access information. Improved bandwidth and new devices can improve reporting and increase the volume of information, but these advances will also create information overload.

An important characteristic of mission execution environments is that people are typically deployed in teams or groups to execute the mission. For example, first responders in disaster areas cannot effectively pursue humanitarian tasks without coordination. Similarly, squads of warfighters in theater must coordinate very closely in order to accomplish missions, perform peacekeeping tasks, or even stay alive.

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 44–63, 2013.

Imagine a scenario where each first responder is given a mobile device with applications and information that will help them execute their mission. What will happen over time is that as first responders move away from the base, the information that is critical is determined by what the individuals and the group as a whole feel, see, hear, smell, or even by situations they cannot directly sense, such as high levels of radiation [1]. In addition, because of the type of work executed by first responders, especially in emergency situations, they are not in position to search for information, or scroll through multiple screens on a device to display the appropriate data when it is needed most. What they need is a capability that can sense as much of the emerging group context as possible, apply that context to share data with the group, and filter data such that only the most relevant information is shared and displayed.

The work presented in this paper is a reference architecture for group-context-aware mobile applications that enables the integration of contextual information from individuals, nearby team members, and potentially the enterprise to support a team executing a mission. Specific innovations of this work include consideration of a wide range of contextual information, including the dynamics of a group operating to achieve a common mission goal. In order to better interact with collaborators and quickly incorporate technological advances, the architecture is highly extensible to support changes in context data models, context data storage mechanisms, context reasoning engines and rules, sensors, communication mechanisms and context views. Section 2 introduces the concept of group context awareness as related to mobile applications at the tactical edge. Section 3 presents the reference architecture. Section 4 presents the architecture decisions and tradeoffs to support extensibility. Section 5 presents a prototype implementation for task management on the Android platform that validates the reference architecture. Section 6 provides a summary of related work. Finally, Section 7 presents conclusions and future work.

2 Group Context Awareness

There are many definitions of context related to context-aware applications [2][3][4][5][6][7][8][9][10][11][12]. Based on a synthesis of available definitions, we define context as any information that can be used to characterize an entity — person, place, or object — such as its properties, behavior, and surrounding environment. We define a context-aware application as an application that uses contextual information to modify its behavior, adapt its user interface, or filter data accordingly.

Most commercially-available context-aware mobile applications (apps) are based on context expressed mainly as location and time of an individual device plus the device user's preferences or history. For example, an app recommends a list of restaurants close to the current location of a user and orders them according to user cuisine preferences combined with the type of cuisine selected in the past by that user.

The guest editor introduction to a recent special issue on context-aware computing presents a challenge for context-aware system developers to work beyond search and location-based services to consider a larger set of context entities in order to improve

their value [13]. Consistent with this statement, the work presented in this paper extends context beyond location and time of an individual user to consider the context of the group that the individual is a part of (e.g. a rescue team). A group-context-aware mobile app first considers individual user context and then relates that information to the group context, thereby helping users understand both their own state as well as the state of the group in which they participate. Desired capabilities of group-context-aware mobile applications in hostile environments include

- Capture and store context information on a mobile device in a non-intrusive manner to reduce cognitive overload and without imposing an unreasonable burden on handheld device resources
- Disseminate context information to group members using whatever communications mechanisms are available at the moment
- Integrate local and group context information to improve mission effectiveness by only sharing and displaying information that is relevant to the individual and mission according to configurable rules

The following section presents a reference architecture for implementing group-context aware mobile applications that enables these capabilities.

3 Reference Architecture

The development of any software architecture should start with a definition of business drivers [14]. Given the early stages of the research project, as well as the fast speed at which technology is changing in the mobile space, we defined the following business drivers

1. Opportunistic integration of new technology
2. Ease of integration with components produced by collaborators
3. Applicability of architecture to different edge-enabled applications

To meet business drivers we defined extensibility as the main architectural driver, expressed as eight scenarios, as shown in Table 1. A sample scenario description, documented according to [14], is shown in Table 2.

Table 1. Extensibility scenarios

#	Name	Attribute Concern
1	Add a new sensor	Separation of concerns
2	Add a new sensor	Modifiability
3	Add a new communication mechanism	Separation of concerns
4	Add a new communication mechanism	Modifiability
5	Add a new content event/action	Separation of concerns
6	Add a new content event/action	Modifiability
7	Add a new context view	Separation of concerns
8	Add a new context view	Modifiability

Table 2. Scenario 3: Add a new communication mechanism

Scenario	Add a new communication mechanism	
Attribute	Extensibility	
Attribute concern	Separation of concerns	
Scenario refinement	**Stimulus**	Developer
	Stimulus source	Developer identifies a communication mechanism that can be used to share context data with other mobile devices
	Environment	Developer is sufficiently comfortable with application to make changes in a reasonable amount of time
	Artifact	Communications Manager of the context-aware application
	Response	Communications Manager is changed to implement message passing using the new communication mechanism
	Response measure	Aside from communication-mechanism-specific code, only the Communications Manager is changed to accommodate the new communications mechanism

The reference high-level architecture for group-context-aware mobile applications is a layered architecture as shown in Fig. 1. The architecture follows the basic architecture for context-aware mobile applications proposed in [2] that divides the architecture into context capture, context reasoning/aggregation and context visualization. This architecture also follows the common model-view-controller (MVC) pattern. The model is the *App Data* in the I/O layer, the controller is the *Application Layer*, and the view is the *User Interface Layer*.

3.1 User Interface Layer

The *User Interface Layer* is the collection of views of context data. The views register an interest in events produced by the system and display data accordingly. The views can also input context data from the user.

3.2 Application Layer

The *Application Layer* is the core of the system. The components in this layer are responsible for managing context and creating events based on individual and group context.

The *Application Manager* is the central hub for all system activity.

The *Context Engine* is the central processor for all context information used by the application. As device sensors report new data and context data is received from

group members, all data is passed through the engine so that new events are detected as they occur. Events are sent to the *Application Manager* for distribution to components that are interested in the events.

The *Sensor Manager* accepts data from sensors on the mobile device, such as position sensors, movement sensors, light and proximity sensors, etc. The *Sensor Manager* also controls sampling rate and change thresholds (the minimum variation in value to report a change) for each sensor.

The *Communications Manager* acts as the gateway for all external communications. Any messages to and from other devices are passed through the *Communications Manager*. It supports multiple communication mechanisms.

The *Data Manager* performs all CRUD (create, retrieve, update, delete) operations on context data and app data, and manages all access to the sensor configuration file and the context rule sets.

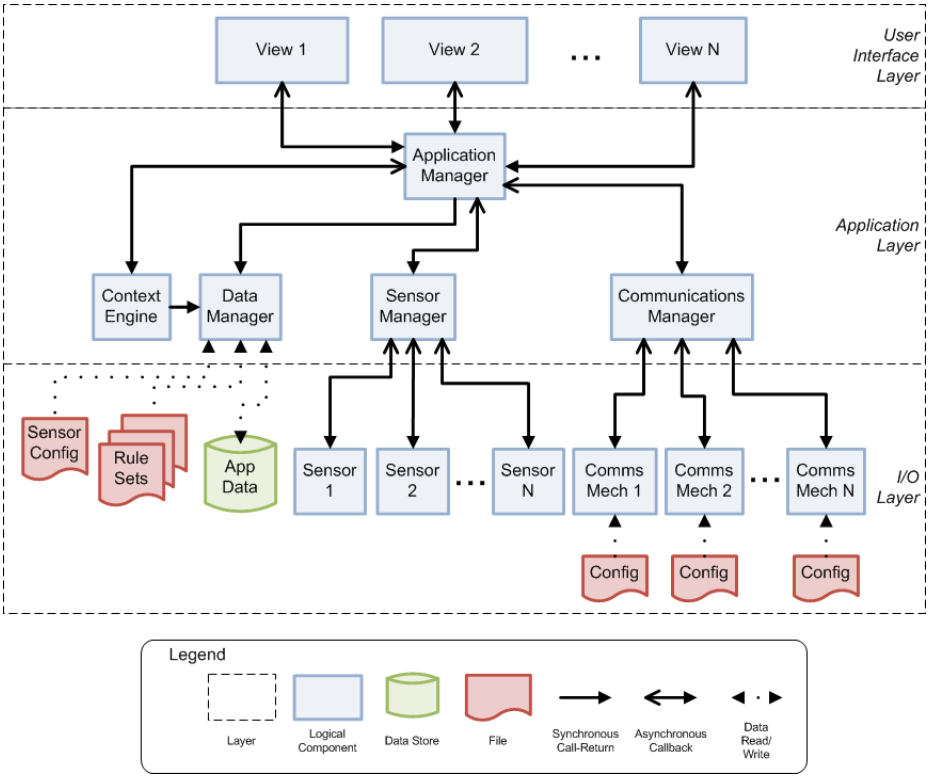


Fig. 1. Reference architecture for group-context-aware mobile applications

3.3 I/O Layer

The *I/O Layer* contains components that interact directly with device I/O elements such as files, databases, sensors and communication services.

Sensor Config is a file that contains default sensor configuration information for all active sensors such as sampling rates and change thresholds.

Rule Sets are files that contain rules that the context engine reasons about. There is a default rule set that is generic to all group-context aware mobile applications. Mission-specific rule sets can be created, added and swapped as needed.

App Data is the physical storage for the context model and app-specific data.

Sensor 1 to *Sensor N* correspond to the components that receive data from sensors. They all implement the same interface so that sensors can be easily added to the system.

Comms Mech 1 to *Comms Mech N* correspond to the communication mechanisms that are used to send to and receive data from other members of the group. They all implement the same interface so that communication mechanisms can be easily added to the system. Each communication mechanism has a configuration file that corresponds to communication-mechanism-specific information such as local addresses, server/router addresses, predefined user names-device pairings, ports, and security keys.

4 Architecture Decisions

There were several architecture decisions that were made to support the required capabilities listed in Section 2 and the extensibility scenarios presented in Section 3. Even though some of these decisions were made using Android-specific programming constructs and technologies, we argue that they can be implemented using equivalent technologies on other platforms.

4.1 Context Model "At the Center"

Given the need to support easy addition of sensors, communication mechanisms, events, and views, a decision was made to place the context model "at the center." This means that the context engine, views, sensors and communication are all based on producing and consuming context data defined by the context model as well as events that are generated based on changes in context data.

The goal established for the context model was to be generic and extensible in order to handle a wide range of situations, environments, and data.

Logical Data Model. The logical data model, or form and structure of the context model, is based on the definition of context provided in Section 2. This work expands the definition of an entity — originally stated as a person, place or object — to include three group-related entities: people (individuals, groups, and organizations), activities, and events. The high-level context model is shown in Fig. 2.

People. The novel contribution of this work is the expansion of the scope of context from the user to the group. As mentioned earlier, in tactical settings individuals rarely

work alone. In most cases, groups of varying size collaborate on tasks to achieve group-level goals (i.e., missions). Therefore, supporting this coordination requires effective sharing of context data between group members.

In the proposed group-context model, the Person entity is changed to People and divided into three subcategories: *Individual*, *Group* and *Organization*. This allows more fine-grained control over how to process each subcategory. Similarly, this breakdown allows further decomposition into different types of groups or organizations.

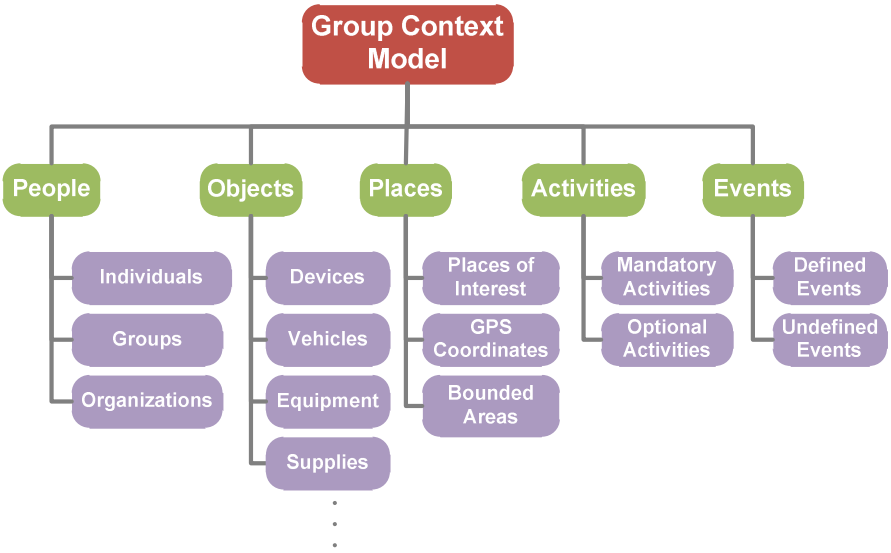


Fig. 2. Group context model

Activities. Activities are what individuals are doing, either on their own or as part of a group. Activities have status (complete, not started, in progress, on hold, etc.) that can be used to model task flow, react to changes in activity status, suggest or assign new activities, and provide information relevant to current activities.

Mandatory Activities are activities that are assigned to an individual and that must be completed (i.e., a task). *Optional Activities* are activities that an individual is performing that do not necessarily require status reporting. This differentiation can be used for tracking activities in a group that are part of a larger mission.

Events. Events are notifications related to changes in context data. Defined Events are set, known events that can either be detected programmatically because of changes in context data that generate the events, or input by users as an external event that can only detected by humans. The application has pre-defined responses for these events. For example, views can subscribe to these pre-defined events or the application may determine that a certain event has to be communicated to other members of the group.

Undefined Events are random events that can only be identified organically by users, and do not have pre-defined responses (e.g., a falling building). As such, they are considered to be informational only and users must respond to them manually.

Physical Data Model. Two options were considered for persisting context data: tables using a standard SQL-based approach, and objects using an OODB (object-oriented database) or ORM (object-relational mapping) approach. This is an important decision because it affects a number of quality attributes, including data model extensibility, performance, power consumption, and scalability.

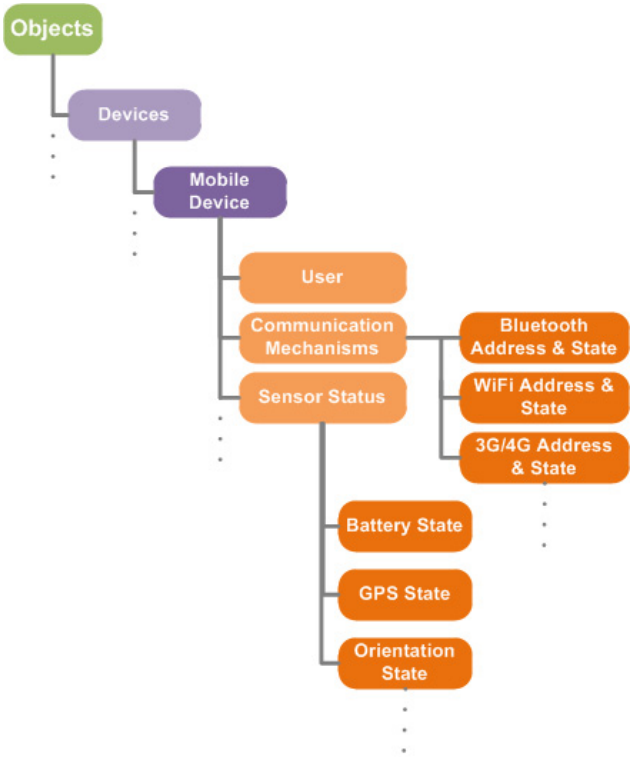


Fig. 3. Context model decomposition for *Devices*

Data model extensibility is better promoted by an OODB/ORM approach because tables do not need to be created or modified in order to support new types of context data. An OODB/ORM approach would either automatically create tables or eliminate them altogether, limiting the effort of extending the data model to that required to add the new context data element within the application. An example of an additional level of decomposition of the context model is shown in Fig. 3. Adding new devices would be a matter of adding a new subclass of Devices. All existing methods that operate on *Devices* would be applicable to the new class of device as well.

Performance and power consumption can be considered together because tests show that they scale together. In performance testing[1], results showed that an OODB implementation (Perst on Android [15]) performs orders of magnitude faster in most tests than an SQL-based implementation (SQLite on Android [16]), as shown in Fig. 4. As a result, less power is used by the OODB implementation, resulting in a correlation between increased performance and reduced battery consumption, as shown in Fig. 5.

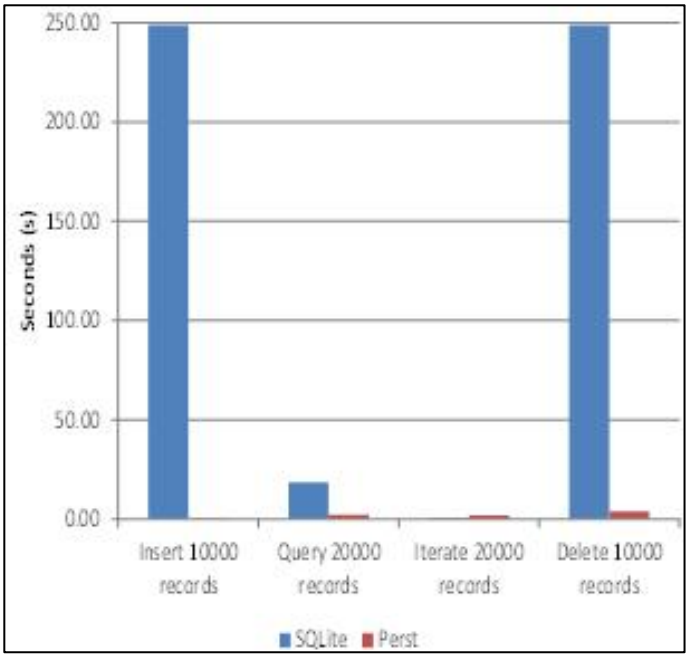


Fig. 4. Performance testing results for SQLite and Perst

The decision was therefore to implement the physical data model using an OODB because of the importance of supporting data model extensibility, given the expectation that the data model will change and evolve for the unique and varied situations first responders and warfighters might encounter at the tactical edge. Similarly, battery power is considered to be a key quality attribute at the tactical edge due to limited resources. The main tradeoff in this decision is scalability because in an OODB implementation most of the data is held in memory. Memory scalability is a concern because a large amount of data is captured by the mobile device sensors and shared between members of a group. The strategy for managing this concern is to limit data capture by disabling or throttling sensor usage as needed and limiting data sharing between group members to only relevant data and events.

[1] Tests were executed on the Android 2.3.4 platform with records of size 172 bytes.

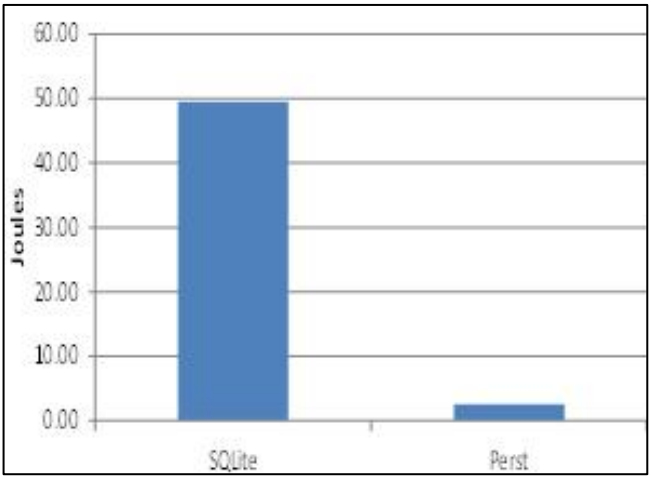


Fig. 5. Energy consumption for 1000-record benchmark

4.2 Context Sensors

To most effectively leverage context, it is important to capture as much of it as possible. As such, the architecture must not only support all available sensors currently onboard the mobile device, but also addition of new ones. To promote the extensibility scenarios related to sensors shown in Table 1, we adopted two architectural tactics: standardized interfaces and decoupling.

Mobile devices currently support a variety of sensors, from light and motion, to orientation, location, and proximity. As mobile devices grow more powerful and hardware components for new sensors become smaller and can be integrated into mobile devices, there will be a continuing need to integrate the inputs from new sensors into the application. To minimize the amount of effort to do so, we designed a standardized interface that any onboard sensor can use to report data to the application, as shown in Fig. 6. This limits the application effort to integrate a new sensor to the development of the sensor control logic (a single file), a few lines of code in the application itself (to add the new sensor and sensor value type, if previously undefined), and a new line in a configuration file (to set sensor defaults). We recognize, however, that as the number of sensors increases on mobile devices, so does the demand on device resources such as CPU, memory, and bandwidth if all sensors are capturing data. Therefore, we employed two approaches to limit the impact of sensor status reporting.

First, we implemented a sensor management capability to enable, disable, and throttle sensor status reporting as needed. This capability can be invoked from a user view or as needed by the application (e.g., based on context rules). Second, we

implemented a strict decoupling paradigm. All sensors are started as Android Remote Services (bound services) [17], which act as entirely separate background "applications" with their own memory space. This means that the processes gathering sensor information are decoupled from the main application while following Android's defined service life cycle [18]. Therefore, even if one or more sensor services are stalled due to high capture rates, the Android process management infrastructure will constrain the impact to the service and limit the resource impact on the primary application. In addition, it enables multiple consumers to use the same source of sensor information leading to an increase in flexibility in the layers above.

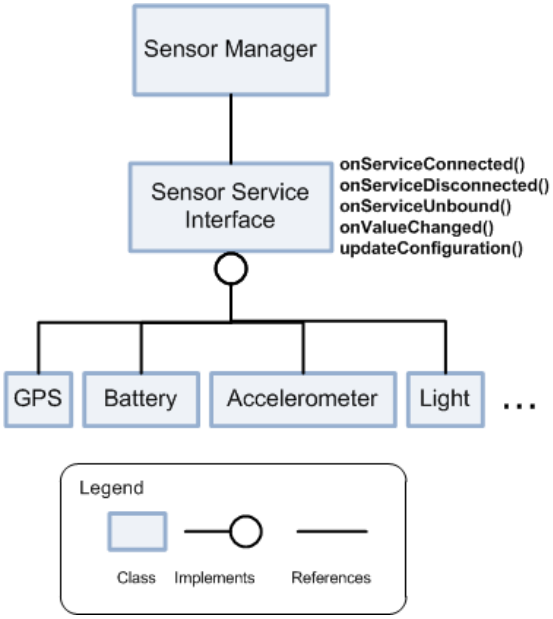


Fig. 6. Sensor service interface

The tradeoffs of this decision are added complexity, increased resource utilization, and performance overhead. Inter-process communication is done using an AIDL (Android Interface Definition Language) interface [19], which is an extra layer of infrastructure. This inherently increases the complexity of the system, making it harder to maintain. It also increases resource usage because the operating system is spawning an entirely separate process for each sensor. However, testing of the prototype with large workloads has shown performance and memory usage to be within acceptable parameters. With respect to maintainability, while this is a legitimate concern as the application evolves, we consider it to be a less important quality to promote than extensibility given the architecture drivers.

4.3 Context Dissemination

As important as it is to gather contextual data, group-context-aware mobile application requires robust context sharing capabilities. Because at the tactical edge it is difficult to predict what communication infrastructure will be available, the goal of the architecture is to support any available communication mechanism in a standardized way.

The architecture decision for enabling addition of new communication mechanisms is very similar to the architecture decision that supports adding new sensors: a standardized interface to easily integrate new communication mechanisms as they become available. The common service interface shown in Fig. 7 provides generic communication methods and callbacks that individual protocols can adapt as necessary. This approach enables the application to use any available communication channels as needed. If multiple channels are available, context information can be used to determine which channels are used and how.

The tradeoff of using a standardized interface for all communication mechanisms is that it implements the lowest common denominator of communication mechanism capabilities. For example, a connection-based protocol establishes communications and controls data transfer in a very orderly, systematic way. A connection-less protocol is much more ad-hoc and cannot be restricted to a specific sequence of events. Therefore, the interface has to support a very general connection process that can allow any sequence of communication events.

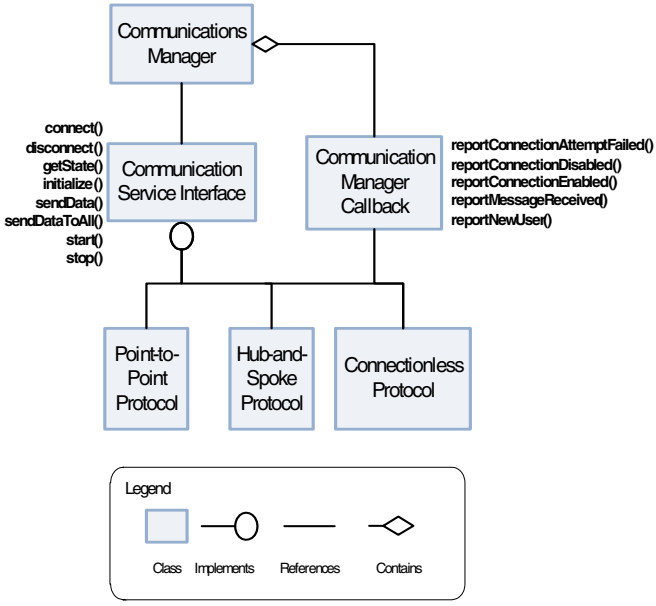


Fig. 7. Communication service interface

Another tradeoff is that in order to intelligently share context data, the application requires the ability to send data to specific users, regardless of the underlying protocol. For connectionless protocols, this could be difficult to support because the underlying protocol may have no knowledge of which user or device is the source of a particular message. It may also lose track of the message target status and actual data might get lost because message transmission is not guaranteed.

4.4 Context Engine

A challenge for any context-aware application is that for any given user, the relevance of any piece of contextual information entirely depends on the situation of the user. For example, the contextual information that a person cares about while at work is significantly different than the information they care about while they are at home, on vacation, playing sports, spending time with family, etc. Therefore, in order to construct a general-purpose group-context aware application, its architecture needs to enable the application to easily switch between different situations so that the relevance of a given piece of contextual data can be tailored to the user as well as the situation that the user is in.

A decision was made to encode all rules in external XML files (rule sets). The set of rules that the context engine reasons about can be changed by loading a different rule set, either at design time or runtime, the latter reflecting the reality of changing situations. A sample rule is shown in Fig. 8.

```
<Rule
 RuleName = "LocationUpdate"
 TriggerDataType = "GPSSensorDataUpdateEvent"
 Conditionals = ""
 Actions = "DECLARE PrimaryDeviceLocationUpdateEvent"
/>
```

Fig. 8. Sample context rule based on a location update event

Each rule in the rule set is evaluated only against a specific type of data item (or subscription object). The triggering data type is specified in the rule by the *TriggerDataType* field. Thus, when a data item is passed to the context engine for processing, only rules that are triggered by the data type of the data item are executed.

Evaluation of the rule is based on the evaluation of conditional statements included in the *Conditionals* field of the rule. The convention of a simple, semi-colon-delimited string is used so that rules can contain an arbitrary number of conditionals in any combination. Evaluations currently supported by the engine include the standard mathematical operators >, <, >=, <=, =, and /=. Conditionals support

comparison to a specific value, written in the rule as a number, or to a field of another object, such as the object that triggered the rule.

The last part of the rule is the *Action* field. There are different types of actions that an application could take if a rule is evaluated to be true based on the receipt of a new data item. For example, the application might need to send data to one or more users in the current group. Alerts might be displayed to users based on the criticality of the information. It is also possible to change the current active rule set so that the application can react to significantly different situations.

As an example, the rule in Figure 8 is called *LocationUpdate* and states that if a local location update is passed to the application by the onboard GPS sensor (this is indicated by the receipt of a data item of type *GPSSensorDataUpdateEvent*) then in ALL situations (no conditionals are evaluated) the application should declare a new event to indicate that the location of the device has changed. The action is that a *PrimaryDeviceLocationUpdateEvent* object is created, containing the new GPS coordinates, and this data is forwarded to all external connections (the encoding of the action is included at the end of the rule set file).

Another example of rule that can be expressed using the same XML structure is shown in Fig. 9. In this rule, called *BatteryLow*, the battery sensor reports that there has been a change in battery level by triggering a *BatterySensorDataUpdateEvent*. The rule looks at the *batteryCharge* field of this event to check if it is less than (*LT*) *30%*. Of this is the case, two actions are taken: one is to create an *ALERT 1* and the other is to declare a *BatteryLowEvent* (as with the previous example, the encoding of the action as well as the alert is included at the end of the rule set file). The full context rule notation and context rule engine implementation is the topic of an upcoming paper.

```
<Rule
 RuleName = "BatteryLow"
 TriggerDataType = "BatterySensorDataUpdateEvent"
 Conditionals = "TRIGGER.batteryCharge LT 30"
 Actions = "ALERT 1; DECLARE BatteryLowEvent"
/>
```

Fig. 9. Sample context rule based on a change in battery level

5 Prototype Implementation

To demonstrate the validity and extensibility of the reference architecture, we built a prototype application on the Android platform that implements a group context-aware mobile application for first responder task management. The concrete architecture is shown in Fig. 10.

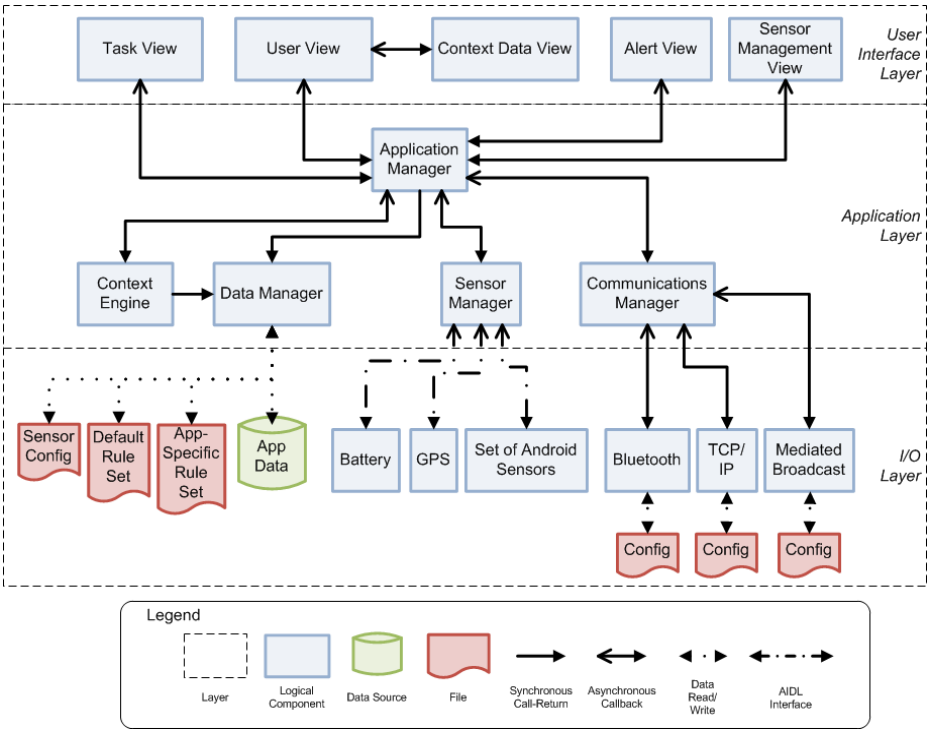


Fig. 10. Architecture for first responder task management prototype implemented on Android

5.1 Views

All views are implemented as Android activities. There are currently three views that we envision would be found in any group-context-aware mobile application:

- *User View*: Displays the list of users that are currently part of the group, as well as any relevant status or other contextual information. This view subscribes to events that indicate user list changes, such as connection events or group addition/modification events.
- *Context Data View*: Displays all relevant context data for a selected member of the group. It is invoked by the *User View* when a user is selected. Data for this view is retrieved from the *App Data* store.
- *Alert View*: Displays alerts to the user. This view subscribes to alert-reporting events.

The *Sensor Management View* was created as part of the prototype to enable manual adjustment of sensor sampling rates and understand the effects of changes in sampling rates. It also serves as an example of a view that changes context data that could also be changed programmatically by a rule (e.g. reduce sampling rates for all sensors when battery <=20%).

The *Task View* is an application-specific view. Groups of first responders are commonly task-based. That is, a leader of the group assigns a task to members of the

group and members report on the status of the tasks assigned to them. This view displays task-related information and responds to task-related events, such as new tasks or task status changes.

5.2 Sensors

Battery, *GPS*, and the *Set of Android Sensors* [20] were added by implementing the *Sensor Service Interface.* The context data elements that map to the information coming from these sensors were added to the context data model. Sensors report changes to their corresponding context model elements. The effort required to add new sensors proved to be as expected (see Section 4.2).

5.3 Communication Mechanisms

Bluetooth and *TCP/IP* communication was added by implementing the *Communication Service Interface*. To test the extensibility of the architecture and its applicability to connectionless protocols, we successfully integrated a *Mediated Broadcast* protocol implemented by research collaborators at George Mason University (GMU)[2]. Effort was within the parameters defined by the corresponding scenario.

5.4 Context Engine and Rules

Tasks were created as *Mandatory Activities* in the context model. *Mandatory Activities* have six fields/attributes: *name*, *participants*, *sub-activities*, *status*, *assigners*, and *assignees*. In addition to the *Default Rule Set* that applies to the three default views (*User*, *Context Data*, and *Alert*), a rule set was created for task management to react to task creation as well as changes in task status (*App-Specific Rule Set*). As new tasks were created these were assigned and sent to appropriate group members. As task status was updated, events were triggered to communicate status updates. An example of an app-specific rule for this application is to evaluate the status of any parent activities of the sub-activity that was completed: if all sub-activities are complete, then set the state of that parent activity should be set to complete as well.

6 Related Work

There is a large amount of work related to architecture and design of mobile context-aware applications. The most-referenced publication in this area is by Dey et al [2]. This work presents a *Context Toolkit* that supports common features required by

[2] The GMU Mediated Broadcast is a location-based ad hoc networking mechanism. It was selected to accommodate situations in which messages are targeted at a specific location rather than an individual (e.g., alert everyone to a disaster situation coming to the area). The mobile devices along the path serve as re-transmitters for the message. This work will be published in an upcoming joint report.

context-aware applications: capture and access of context, storage, distribution, and independent execution from applications. Our work extends the definition of context to include group-related entities as well as the capabilities of the main components of the toolkit (widgets, interpreters and aggregators) to process this group context. It also adds a context dissemination component to share context with group members. Baldauf et al [10] conducted a survey of multiple approaches for architecting context-aware systems and determined that it largely depends on how the system captures context data. It also claims that separation of detecting and using context is necessary to improve extensibility and reusability of a system. The proposed reference architecture is consistent with these two statements. Most other research in this area presents ideas and frameworks for designing context-aware systems, but is limited to individual context [3][7][9][12][24][25]. For example, Cagalaban and Kim [9] present an architecture of a context-aware system that includes web services, mobile devices, smart homes, and the different responsibilities in a health care context. The use of a rule engine is discussed for processing context in an individual system. As another example, Henricksen and Indulska [12] propose a notional layered architecture that uses separation of concerns, synchronous and asynchronous communications, and other architectural constructs to manage the capture and processing of contextual information in an individual system setting.

Commercial, location-based apps running on smartphones can filter information based on a user's current position, but do not use other information useful in tactical environments such as the state of individuals or their devices. Work in the commercial sector is just beginning to use richer context information from multiple individuals to inform handheld users and make recommendations (e.g., finding a possible restaurant for a group lunch) but do not treat the group as a collaborating resource. In addition, as with the related research, context is mainly limited to location and interests as part of social networks. Examples include SocialFusion [21], CenceMe [22] and Serendipity [23].

7 Conclusions and Future Work

Handheld mobile technology is reaching first responders and soldiers in the field to help with mission execution. Using this technology to sense as much of the environment as possible, share this information with members of a group, and reason about the group context can improve the effectiveness of the mission and optimize resources. In this paper we have described a highly-extensible reference architecture for group-context-aware mobile applications that integrates the contextual information from individuals with that of nearby team members.

Context is extended beyond the location and time of an individual to consider group context elements such as activities and events. A variety of sensors and communication mechanisms can be integrated into the architecture to account for the variability in devices as well as communication mechanisms that are available. A context engine based on mission-specific rule sets enables applications to reason about context and make decisions on what information to show to individuals, what

information to share with the group, and optimize individual and group resources. The next step is to implement a disaster recovery scenario and corresponding rule set in which resource-poor first responders are able to coordinate their efforts by sensing and sharing context information that is automatically filtered based on mission parameters. Once the scenario implementation is stable we will attend experimentation events where we will be able to test the architecture in a simulated setting and see how the architecture behaves under close-to-real infrastructure and data rates.

One of the more interesting results of this work has been the ability to leverage the architecture to support collaboration. By identifying extensibility scenarios early on in the design process, we were able to construct an architecture that supports multi-organizational collaboration to construct and evaluate different pieces of the architecture. This has allowed us to reach out to researchers from multiple universities and industry, resulting in synergistic research and development, furthering the goals of all participants.

Current and future work includes:

- Group context-aware resource optimization: use group context information to optimize battery consumption, bandwidth and computation resources of a group. Examples include the use of device status to assign computation-intensive tasks to devices with the most resources; adjustment of sensor input rates depending on battery level; and selection of appropriate communication mechanisms for context dissemination depending on comms status, information sensitivity and battery level.
- Minimizing user interaction for context capture: exploit device sensors to capture context information in a non-intrusive manner such that there is minimum disruption to the user's attention and activities. An example includes the use of the accelerometer.

Acknowledgements. This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. This material has been approved for public release and unlimited distribution (DM-0000029).

References

1. Alabaster, J.: Japan's Softbank to offer world's first phone with radiation detection (2012), http://www.itworld.com/278997/japans-softbank-offer-worlds-first-phone-radiation-detection
2. Dey, A.: Understanding and using context. In: Personal and Ubiquitous Computing, vol. 5, pp. 4–7. Springer-Verlag London Ltd. (2001)
3. Godbole, A., Kim, S.-Y.: User centered design of context aware cell phones in human-centric systems. In: 2010 IEEE International Conference on Information Reuse and Integration (IRI), August 4-6, pp. 189–194 (2010)

4. Chen, H., Finin, T.: An ontology for context-aware pervasive computing environments. In: Proc. IJCAI Workshop on Ontologies and Distributed Systems, IJCAI (2003)
5. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. In: Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, California, pp. 85–90 (December 1994)
6. Schmidt, A., Aidoo, K.A., Takaluoma, A., Tuomela, U., Van Laerhoven, K., Van de Velde, W.: Advanced Interaction in Context. In: Gellersen, H.-W. (ed.) HUC 1999. LNCS, vol. 1707, pp. 89–101. Springer, Heidelberg (1999)
7. Malek, J., Laroussi, M., Derycke, A., Ghezala, H.B.: Model-driven development of context-aware adaptive learning systems. In: Proc. 2010 10th IEEE International Conference on Advanced Learning Technologies, pp. 432–434 (2010)
8. Chen, G., Kotz, D.: A survey of context-aware mobile computing research. Dartmouth Computer Science Technical Report. TR2000-381 (2000)
9. Cagalaban, G., Kim, S.: Context-Aware Service Framework for Decision-Support Applications Using Ontology-Based Modeling. In: Kang, B.-H., Richards, D. (eds.) PKAW 2010. LNCS (LNAI), vol. 6232, pp. 103–110. Springer, Heidelberg (2010)
10. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing 2(4), 263–277 (2007)
11. Weerasinghe, T., Warren, I.: Odin: context-aware middleware for mobile services. In: Proc. 2010 6th World Congress on Services, pp. 661–666 (2010)
12. Henricksen, K., Indulska, J.: Developing context-aware pervasive computing applications: models and approach. Pervasive and Mobile Computing 2(1), 37–64 (2006) ISSN 1574-1192, doi:10.1016/j.pmcj.2005.07.003
13. Mehra, P.: Context-aware computing: beyond search and location-based services. IEEE Internet Computing, 12–16 (March/April 2012)
14. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice, 2nd edn. Addison Wesley SEI Series in Software Engineering (2003)
15. McObject, Embedded Database for Android – Perst Small Footprint DBMS (2012), http://www.mcobject.com/android
16. SQLite.org, SQLite Home Page (2012), http://www.sqlite.org/
17. Android Developers, Bound Services | Android Developers (2012), http://developer.android.com/guide/topics/fundamentals/bound-services.html
18. Android Developers, Services | Android Developers, (2012), http://developer.android.com/guide/topics/fundamentals/services.html#Lifecycle
19. Android Developers, Android Interface Definition Language (AIDL) | Android Developers (2012), http://developer.android.com/guide/developing/tools/aidl.html
20. Android Developers, Sensor | Android Developers, (2012), http://developer.android.com/reference/android/hardware/Sensor.html
21. Beach, A., Gartrell, M., Xing, X., Han, R., Lv, Q., Mishra, S., Seada, K.: Fusing mobile, sensor, and social data to fully enable context-aware computing. In: Proceedings of the Eleventh Workshop on Mobile Computing Systems Applications (HotMobile), Annapolis, MD (February 2010)
22. Miluzzo, E., Lane, N., Fodor, K., Peterson, R.A., Lu, H., Musolesi, M., Eisenman, S.B., Zheng, X., Campbell, A.T.: Sensing meets mobile social networks: the design, implementation and evaluation of the CenceMe application. In: Proc. of 6th ACM Conference on Embedded Networked Sensor Systems (SenSys 2008), Raleigh, NC, USA, November 5-7 (2008)

23. Eagle, N., Pentland, A.: Social serendipity: mobilizing social software. IEEE Pervasive Computing 4(2), 28–34 (2005)
24. Li, X., Lin, J., Li, L.: On the design of a mobile agent environment for context-aware m-commerce. In: 2010 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), July 9-11, vol. 3, pp. 176–180 (2010)
25. Hsu, H.J., Wu, S.Y., Wang, F.J.: Methodology to developing context-aware pervasive applications. In: Proc. Fifth IEEE International Symposium on Service Oriented System Engineering, pp. 206–213 (2010)

Mobile Apps Development: A Framework for Technology Decision Making

Emiliano Masi[1], Giovanni Cantone[2],
Manuel Mastrofini[2], Giuseppe Calavaro[3], and Paolo Subiaco[3]

[1] FIZ Karlsruhe, Hermann-von-Helmholtz-Platz,
76344 Eggenstein-Leopoldshafen, Germany
emiliano.masi@fiz-karlsruhe.de
[2] University of Roma Tor Vergata, DICII
Via del Politecnico, 1 – 00133 Roma, Italy
{cantone,manuel.mastrofini}@uniroma2.it
[3] IBM Italia SpA
Via Sciangai, 53 – 00144 Roma, Italy
{gcalavaro,paolo_subiaco}@it.ibm.com

Abstract. Developers of a new Mobile App have to undertake a number of decisions, including the target platform and the development technology to utilize. Even though there is no one-size-fits-all solution, which could meet all needs for all contexts, this paper is concerned with an exploratory study aimed to provide developers with a framework to support their technology selection process, including practical guidelines on how to select the technology that best fits the given context and requirements. The exploited research methods are survey, interview, and case study. Results consist in a model of, and a collection of data and experts' experiences about, some advanced platforms. Results are packed in a tool-prototype: once entered the needs and required device features, the tool returns measures that allow a decision maker to identify the development technology, among the recommended alternatives, which best fulfills the actual requirements.

Keywords: Mobile App development, mobile platforms, mobile development technology, mobile development approaches.

1 Introduction

The basic question that this paper tries to answer is: What is the right technology to use for the development of a mobile application for given requirements and context?

In order to develop a new Mobile App, in fact, several decisions have to be made. The first one is the platform on which the application will run. Such decision is usually made by the marketing management. Very often, more than one platform is chosen, although versions for the different platforms can be released at different times. Instances of platforms include, but are not limited to:

- iOS platforms,
- Android platforms.

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 64–79, 2013.

As soon as the platforms decision is undertaken, one more very important choice is required, which matches the aforementioned question: What technologies should be used to develop the given mobile app? Now, the Technical Leader has the ownership of such decision. As we will be explaining in the next sections, the most known examples of such technologies include:

- HTML5;
- CSS;
- JavaScript;
- Platform Specific Software Development Kit (SDK), including:
  a) Android SDK, and
  b) Apple® XCode;
- Titanium Mobile, and related supports for Mobile App development;
- Adobe®, and related supports for Mobile App development;
- PhoneGap;
- jQueryMobile;
- Sencha Touch;
- DojoMobile.

Indeed, even when a single platform is initially targeted to develop an app, few ways exist to enact the development, which can be classified in three Development Approaches (DAs):

- Native DA: i.e., pure native development, using the platform specific SDK;
- Web DA: i.e., pure web app (HTML 5 allows Apps that are almost as powerful as native apps);
- Hybrid DA: i.e., using a mixed approach, as supported by several different technologies discussed in the remaining of this paper.

A decision concerning the DA to use has to take into account both functional and non-functional app requirements, such as the need for different devices on board (GPS, Camera, etc), as well as constraints placed by the selected target platforms, schedule, and budget. Therefore, a multi-criteria decision process is required [1][2], and both tools and frameworks would desirable to support and guide it.

2 Purposes of the Study and Research Methods

Evidence-based research in the field of software development for mobile apps is in its beginning and still needs enhancement. Data and knowledge bases are not yet publicly available. Concerning private companies, it may be that data collections are not yet capitalized organization-wide. As a consequence, it is still frequent for researchers to start studies by producing their own data from scratch.

The research methods that we used for this study are survey and case study research. For data collection, we used literature and technical documentation survey, interviews to practitioners, and development of case studies by using and comparing different technologies.

The purpose of this work is exploratory rather than confirmatory [3]; in fact, we do not aim to give a final answer to the basic research question, but we want to check whether an approach is feasible and capable to support answering the basic question. Consequently, in the remaining, no formal goals [4], hypotheses [5] or details concerning the utilized research objects and processes (case studies, handbooks, papers, etc.) will be presented.

In general, we do not aim to address the false problem of defining the "absolute best" technology for mobile software development. Vice versa, we remark that our expectation is that no single development approach will fit all needs of all customers. As Fling observed [6], whatever medium we use to address our goals, we have a number of choices, each with its own pros and cons. Similarly to all other branches of Software Engineering, we expect some mobile media to be quick to create apps, but accessible to a few of the customers and/or delivering hard to maintain apps; others could address a larger market, but far more expensive and complex to use.

However, it is clear that companies are facing an obvious trade-off between user experience and application functionality on the one hand, and development costs and time to market on the other hand. Therefore, the challenge is to choose a development approach and a technology that are capable to balance requirements with the available budget and time-to-market [7]. At this point, the fundamental question is: "What is the right choice?".

In order to help answer such question, this paper aims to provide

(i) a guide to technology decision, and
(ii) a framework to support such decision, which could evolve based on new technologies that will come out.

3 Characterizing the Technologies to Study

The choice of the platform dictates the range of technologies to use for developing an app. If we need an application expected to be multiplatform, then we have a range of technologies and possible approaches to use - besides developing the same app for each platform using their native SDK, - i.e. hybrid or web approaches. Conversely, if we need an app for a single Operating System (OS), then we have different choices available and the right approach could be the native development.

In the remaining, we will try to answer the initial question about mobile technologies by focusing on three Platform Categories (PCs):

PC1. iOS platforms;
PC2. Android platforms,
PC3. Other platforms, which we merge under this category, namely "Others".

As already mentioned, after the selection of the target platform(s), the next step consists in choosing both the Development Approach, and the Development Technology (DT).

There are three DAs available for apps development:

DA1. Web apps
DA2. Native apps
DA3. Hybrid apps.

Concerning the DT, it is not independent from the selected DA and PC. In this paper, we take into consideration a limited set of development technologies, among the ones available on the market. Specifically, we consider those technologies that the largest software companies commonly adopt worldwide. We have already listed such technologies in the previous Section 1. In the following, we consider these technologies and group them by the selected DA.

Web apps: choosing the technology is quite straightforward:

DT1. HTML5, CSS, and JavaScript. Since these technologies fit together, we consider this set as a single choice.

Native apps: the platform specific development kit can be selected, e.g.:

DT2. Android SDK;
DT3. Apple XCode.

Additionally, the following technologies can support the development of Native Mobile Apps:

DT4. Appcelerator®'s Titanium Mobile,
DT5. Adobe® Flex, in conjunction with Adobe® Air.

Last two development technologies can create cross-platform apps from the same source code.

Hybrid apps: PhoneGap can be used, in addition to the following JavaScript frameworks:

DT6. (PhoneGap) + jQuery Mobile;
DT7. (PhoneGap) + Sencha Touch;
DT8. (PhoneGap) + Dojo Mobile.

A brief review of the listed technologies, including their pros and cons, is provided in the remaining of this Section.

3.1 HTML5 + CSS3 + JavaScript (DT1)

Since they are web technologies, the learning curve is expected to be very steep (i.e. we expect developers to quickly get familiar with them), if compared to the native languages. Additionally, when writing an application, multiple platforms can be targeted without encountering any significant problems. On the other hand, since HTML5 and JavaScript are not allowed to access all device features, the developer could encounter some limitations. Moreover, as an application built with such stack has to be interpreted by the browser, performance can be worse than an application developed with native languages.

3.2 Platform Specific Software Development Kit (DT2,3)

It ensures the full exploitation of all features of the platform on which the practitioner decides to develop the app, in addition to the native look & feel. The most obvious downside is the ability to address only one platform; for this reason, if the organization is willing to produce a multiplatform application, the code has to be rewritten for each

chosen platform. This contributes to get high development cost and long developing time. It's important to underline that in the following DT2 and DT3 will be considered together.

3.3 Appcelerator® Titanium Mobile (1.8.1) (DT4)

Titanium Mobile is a cross-platform mobile development framework that enables developers to write JavaScript code, which is then compiled down to native code for the iOS and Android platforms. If the organization wants to build a multiplatform application, Titanium Mobile can save time with respect to developing via platform specific SDK. Additionally, a real native application is obtained, with the platform-specific look & feel.

3.4 Adobe® Flex Mobile (4.5) + Adobe Air (DT5)

Flex is another cross-platform mobile SDK. It enables developers to write their applications by using the ActionScript language for Android, iOS, and Blackberry platforms. Concerning iOS, it generates a pure native application written in Objective C. Concerning Android and Blackberry, the developer has to install a further layer on which the application will run: the Adobe Air Environment. Applications developed by using Flex are deployable on the most important app stores.

3.5 PhoneGap (1.7.0) (DT6, 7, 8)

This technology enables developers to create a web application wrapped into a native container. This means that developers can write an application once, and, when they need to migrate to another platform, they can reuse the code by wrapping the initial web app into another native ad-hoc wrapper. Since both wrappers offer the same interface to developers, no changes are required to the previously written code. Developers still have to face the cons associated to using HTML5, CSS, and JavaScript (see Sub-section A above) but, at the same time, they receive some of their benefits, including a steep learning curve, the possibility of deploying the app on appstores, and the chance of extending the native wrapper to use device features not available in the default HTML5.

Since PhoneGap allows to develop pure web apps, selecting a JavaScript framework appears to be an essential choice. In this paper, we take into account three main JavaScript frameworks, among the ones available on the market, i.e.: jQuery Mobile, Sencha Touch, and Dojo Mobile. jQuery Mobile seems to us to be a good candidate for very basic and logically simple applications, while Sencha Touch, due to the different development approach, has to be preferred when the organization is willing to leverage more complex development patterns, such as the Model-View-Control. It seems to us that this 1^{st} technology shows a very flat learning curve, but it gives practitioners the possibility to keep under management the development of even the more complex applications. Dojo Mobile seems to us to be the most complete technology, among the three considered above, as it provides both development approaches offered by jQuery and Sencha. However, if a simple business logic, small footprint, and very small time to market are needed, jQuery Mobile could be the right choice.

4 Drivers for Technology Selection

In our study, the technology selection is driven by a main set of requirements, grouped by the following two categories:

(i) Needs
(ii) Device features.

Based on literature [6], [7-16], we can breakdown these categories as in the followings.

Needs represent both functional and non-functional requirements. Needs, or, equivalently, Non-Functional Drivers (NFDs) are:

NFD1. Access to native hardware features;
NFD2. High performance;
NFD3. Cross platform;
NFD4. Easily upgradeable;
NFD5. Deployable on app stores;
NFD6. Small footprint;
NFD7. Quick coding and prototyping;
NFD8. Complex
NFD9. Simple business logic;
NFD10. Custom look & feel;
NFD11. Platform look & feel;
NFD12. Cheap Development cost.

Device Features (DFs), as commonly found on mobile devices, include:

DF1. Accelerometer;
DF2. Compass;
DF3. Orientation;
DF4. Light;
DF5. Contacts;
DF6. File;
DF7. Geolocation;
DF8. Media;
DF9. Network;
DF10. Notification Alert;
DF11. Storage;
DF12. Multitouch;
DF13. SMS;
DF14. Bluetooth;
DF15. Video Capture.

5 A Model for Evaluating Mobile Technology

Based on the content of sections 3 and 4, answering to our basic question − What is the right technology to use for the development of a mobile application for given requirements and context? − means to solve a problem in a space made by forty dimensions: 3 related to PC, 3 to DA, 7 to DT, 12 to NFD, and 15 to DF. Three of

these dimensions, the PC-related ones, are alternative dimensions; the same holds for the three DA-related dimensions (this could be used to merge them in two dimensions and downsize the space to 36 dimensions). The twelve NFD-related dimensions are independent one another, and the same holds for the fifteen DF-related dimensions; however, DF dimensions can depend on NFD, DA, and PC dimensions, even though we might still be unable to express, a priori and formally, such dependencies. Additionally, an application is required to meet many needs, a device feature can serve many needs, and each development technology can offer different features, depending on the platform it is used for. In other words, we are not coping with a system that we can immediately and easily model via mathematical functions, assuming it is feasible.

To manage the complexity, our decision was to proceed by abstraction and classification, and we eventually consolidated those dimensions in five macro-dimensions. This led to constructing a five-macro-dimension space of: Needs, Device Features, Development Technologies, Platforms, and Development Approaches. The domain of each macro-dimension includes the items listed in the previous sections for PC, DA, DT, NFR, and DF. Since we see no interest in sorting those items, in their domains, they are listed in arbitrary order, so obtaining a Nominal scale [5], [17], [18] for each dimension.

This discrete space of five macro-dimensions can be easily and automatically generated and updated whenever new platforms or development technologies/ approaches appear on the market or are removed, for any reasons, from the set of accepted candidate technologies/approaches.

Our next step is to represent: (i) each technological stack as a point in this space, according to its capability to meet some requirements (e.g. support for a given device); and (ii) the app to develop as a point in the same space, according to the importance that each requirement has with respect to the app context (e.g. importance of the presence of a given device).

This way, a proximity measure could be created to quantify the distance between the point representing the app and each point representing a technological stack. The technological stack which minimizes the distance from the app is the first candidate to be the best fit to the considered app context. The feasibility of mapping technological stacks and app requirements as points in the previously defined space can follow the impact vector model [19]; in particular, the app requirements can be modeled as a "goal impact vector" (i.e. the expected result of the development), while technological stacks can be modeled as "strategy impact vectors" (i.e. the possible combinations of languages, approaches, platforms and tools that can be used for the development) [19]. For brevity, we omit the details on how to map the app and the technological stacks to points in the space. However, a proximity measure still has to be defined, so that a Tech Expert can apply our model in a real context; a definition of such proximity measure and an example of use of our model are provided in section 8.

6 Implementation and Graphical View of the Model

There are both numerical and graphical problems to represent a space with more than three dimensions. Concerning the first, discrete dimensions and Ordinal scales rather than Real scales are possible for the interesting functions, with consequent limits on

the mathematics that can be applied when looking for optimal solutions under defined constraints. Concerning the last, our decision was to start by using a simple user interface; in fact, the focus of this paper is on (i) exploring the mobile development technology modeling for decision-making; and (ii) empirically verifying models in lab and on the field. This is the reason why we did not put much effort on building up an advanced user interface for our tool prototype. In particular, the prototype is based on a table (see Table 1) which implements the space defined in section 5. In order to model the five dimensions in a flat surface – which allows an easy use of the model, – the item type of such table (i.e. a point in the space) represents more than one piece of information; specifically, each item of the table is itself multi-dimensional, so creating a topological space similar to a manifold [20]. Each elementary item of the table (i.e. each dimension of a given point) expresses a measure in a four-value Ordinal scale, which are graphically represented by the symbols , , , and , where denotes the null value, i.e., "Not supported".

In practice, the table represents a complex but homogeneous abstract entity, i.e. a 5-dimensional hypercube, whose elements are symbols of the given ordinal scale.

Table 1 is structured in two quadrants. These realize two guides for selecting the technology to use according to: a) the needs, and b) the device-specific hardware features that the particular technology offers, respectively.

The first quadrant implements the subspace (DT, NFD, DA, PC); the abscissa reports the development technologies, the ordinate represents the needs. The type of development approach, as leveraged by the corresponding development technology, is represented by different columns and different background colors: (i) Web in the first column (Orange); Native in the subsequent triple of columns (Blue); and Hybrid in the last triple of columns (Green). Each item shows a value in the given Ordinal scale. Since the values reported do not depend on the platforms specificities for each given development technology, such platforms are not represented explicitly. In practice, this quadrant represents a cube with colored slices; each element of the cube includes as many times the ordinal measure, shown in the first quadrant, as the number of platforms.

The last quadrant shares the abscissa (DT) with the first quadrant, while the ordinate represents the device features. Many items in this quadrant represent arrays of platforms (rather than any platform as in the first quadrant). The reason is that each technology can offer different features depending on the platform on which it is used, and a measure is expected for each of those platforms. Again, the actual platforms that are considered separately are iOS and Android, whereas the remaining platforms are grouped into the category "Others".

The scale elements assume a slightly different meaning in the two quadrants. In the first quadrant, the semantic of the scale is {Null, Insufficient, Sufficient, Excellent}; in fact, the symbols indicate the extent to which, for the considered development approach, that specific technology satisfies the corresponding need: insufficiently, sufficiently, and excellently. In the last quadrant, instead, the semantic of the scale is {Null, Rough, Medium, Well}; in fact, those symbols indicate whether the specific technology provides APIs for using the corresponding device feature, and their level of support: "Well", i.e., supported for all the different versions of the same platform, "Medium", i.e., supported for some but not all the different versions of the same platform", or "Rough", i.e. lightly supported. In the first quadrant we can find

only one symbol per table item, whereas in the last quadrant, as already mentioned, an item is an array of symbols (i.e. a 4-dimension point), and can also include all symbols together (one per point dimension).

We filled out the dimensions of the matrix discussed above by collecting information from various scientific papers, personal experiences (first quadrant), and technical online documentation (last quadrant). Table 1 also synthesizes on the results from this work.

7 The Process of Technology Decision Making for Mobile

As already mentioned, we want to develop a guide to select the best technology to use for the development of a specified mobile application in a given context. For this reason we can say that our output is the Technology dimension, while the other dimensions of Table 1 are our inputs.

Probably, a multivariate analysis should be used for identifying the best technology to exploit under the given requirements and constraints. However, at this stage of our work, since there is no way to apply multivariate with the small and artificial dataset available, our decision was to start by using simpler techniques.

In general we have two kinds of requirements: General Requirements (GRs), which are present in any mobile app, and Specific Requirements (SRs), which are explicitly stated for the current app, and are subtypes of the GRs.

In order to enact a technology selection process, the Tech Experts are required to assign a weight to each GR, based on its importance or relevance for any app to develop, and then to give a score to each requirement. Concerning the latter, let N be the number of points to distribute across needs and device features. In the setting that we used for our approach, 25% of such points are automatically and evenly assigned to the GRs, and the remaining 75% of the N points are left for assignment to Tech Leaders; these will assign them as they prefer to the SRs, based on their assessment of the requirement relevance for the app being developed. The initial 25% assignment is made to diversify the obtained results, without losing the contributions that other features give to the decision making. This point distribution can be changed arbitrarily, without affecting the model validity.

Finally, Tech Leaders enter Table 1 and obtain, as a result, a value in the previously defined Ordinal scale for each technology. In order to get more manageable results, they can make the further decisions of translating those values in the notation of Real numbers, e.g., by using a locally defined translation map; this allows to switch from an Ordinal scale to a Real measurement model, which enables to apply the algebra of Real numbers. At this point, each generated number represents a numerical score which should quantify numerically the capability of each technology to fulfill any given requirement.

We are aware that the mentioned scale transformation is a theoretical and practical hazard [17,18]. However, also in this case, the impact vector model supports us to combine and manage heterogeneous dimensions [19]. Since the goal of this paper, as already stated, is not to provide a formal and final model for technology selection, the mathematical foundation of our model, including the scale transformation rationale, is

not herein detailed. Nevertheless, practical evidence of the model validation and a case study are reported in the remaining sections, while the definition of a formal model is deferred to a shortly coming future work.

8 Case Study

As an example of using Table 1 for decision making, let us consider a synthetic version of a case study we conducted.

Let us suppose that a Tech Leader is requested to develop an application with the requirements shown in the Table 2, row 1.

Ten features are explicitly stated as significant for this application, as shown and numbered from 1 to 10 in Table 2, row 2. Let the Tech Leader assign the same weight to all general requirements (GRs), and assuming N=1000, to distribute the remaining 750 points on each specific feature (SRs) in the following way:

1. 100 points, 2. 30 points, 3. 10 points, 4. 60 points, 5. 150 points,
6. 30 points, 7. 80 points, 8. 70 points, 9. 70 points, 10. 150 points.

Additionally, in order to have manageable results, let the Tech Leader map the given Ordinal scale into a Real scale, as explained in the previous section, by using the following map: Excellent: 2.0 points. Sufficient: 1.0 point. Insufficient: 0.5 points. Not Supported: -1.0 point. This gives a Real scale in the range from -500.0 (worst case, i.e. Not supported in every dimension) up to 2000.0 (ideal technology for the given requirements, i.e. excellent in every dimension.)

The Tech Leader can now enter Table 1, and obtain an ordinal score for each technology. Subsequently, he/she can enter the map and translate the ordinal symbols found into Real numbers; eventually the results shown in Table 3 will be obtained.

As we can see, the two technologies with the highest scores are:

1. Platform Specific Development Kit with 1675.9 points.
2. Adobe Flex Mobile + Air with 1491.05 points.

For the given context, the more meaningful definition of proximity measure is the vector distance from the optimal vector (i.e. the resulting difference vector). The optimal point is the one with maximum achievable score, i.e. 2000 points, which exactly matches the required characteristics of our app. In practice, for our context, minimizing such distance is equivalent to pick the maximum among the computed technology scores. Based on the scores shown in the Table 3, the best technological choice for this application should be the native approach with the Platform Specific SDK. A good choice would be also the Adobe Flex Mobile + Air, in fact, if there are heavy constraints on the time to market and there is not enough in-house skills for each platform (which is a characteristic not represented in our model), such technology could be the only feasible way.

In both cases, however, developers are given all the information required to make an informed choice on what technology best fits the application to develop in the given context.

9 Validity Issues, Solution Limitations, and Lesson Learned

The model proposed in the previous sections was populated by interviews with experts of, and tested in four case studies within, the IBM Italy Rome Smart Solutions Lab. Such case studies were concerned with different problems to address and they covered a wide range of possibilities, but in a limited number of domains (e.g., mobile banking, education, and mobile marketing).

Once minded that we are coping with an exploratory study, based on the feedback that we had from practitioners, the solution validity of the study, i.e., its *internal validity*, can be considered high enough; on one side, this is concerned with the solution *usefulness* (does it address relevant practical problems?), which was very positively evaluated by the involved experts; on the other hand, it is concerned with the *goodness* (how well the solution is constructed); in the opinion of the same practitioners, practical and actionable results were produced by the study, even in absence of a strong and theoretically rigorous framework.

In our view, the proposed solution is *portable* to, and promises to be *effective* in, different contexts and cases. In other words, the *external validity* of the proposed solution is considerably high. However, due to the exploratory nature of the study, consequential *limitations* should be taken into consideration.

First of all, it is important to remark that interviews, as source of collected data and knowledge, involved quite a limited number of participants in the role of mobile technology experts. Also, only one subject developed the four case studies that we have been performing. Additionally, the proposed model was piloted with a single organization. This increased the threat of having the same person repeating all positive or negative actions from case to case, and people doing the technology evaluation in the same context.

One more aspect, which was not considered in this paper, and which affects the external validity and limits of the proposed solution, relates to the communication model utilized by the app to develop, e.g., apps connecting more than one partners synchronously and/or asynchronously. This includes considering if, and to which extent, media-content/social networks would influence the development of mobile apps, and how this should be modeled by an enhanced version of the proposed solution.

Again concerning threats on validity and limits of the proposed solution, we notice that our model is oriented to supporting project leaders in decision making, but it is based only on technical attributes. Constraints placed by the strategic management should be introduced in the model, to support technicians to undertake value-based decisions [21], including risk and return on investment, and proceed coherently with the strategic goals of their company.

A further relevant aspect is that the information gathered during interviews was mainly qualitative knowledge; additionally, measures in Ordinal scale, as produced from case studies, were eventually influenced by the subjectivity of the involved experts.

Also, one more threat to internal validity is the lack of a complete formal model supporting the proposed solution. Indeed, the discrete macro-space, which this paper has provided to support decision-making for mobile development, is not a Euclidean space, so, in general, we cannot add scores up as easily as we did in the case study described in section 8. Also, we have not considered interdependent choices of technologies, mutual influences etc.

In our understanding, in case of exploratory studies, which is our case, it is reasonable to accept all the aforementioned validity threats, given the mitigation strategies we enacted, as usual in the experimentation in the field of Software Engineering [5].

For what concerns the lesson learned, let us recall that interviews with experts helped populate Table 1. However, as the case studies proceeded, we had to include additional attributes in the model (i.e. dimensions in the space) and refine the seeded values. This confirmed our conviction that an iterative-incremental approach should be enacted to define the model and populate its base of knowledge. Following the initial training of the data and knowledge base, as enacted by technology experts, the enrichment of the base should be enacted continually, based on decisions made by the organization's Tech Leaders, possibly leveraging a structured methodology, e.g. the Quality Improvement Paradigm [22]. Additionally, each development organization might need to manage its own base of knowledge for mobile apps. These solutions should also help solve expected dependencies of the model from the context of the organization, e.g., by realizing organization-wide dynamic bases of knowledge refined by domain (mobile health, mobile education, etc.).

10 Previous Works

A basic book concerning Mobile was authored by Brian Fling and published on 2009 by O'Reilly Media [6]. This book includes the fundamental concepts and techniques for creating mobile sites and apps.

On market side, in the year 2010, the Vision Mobile web site examined the latest insights in the mobile market, provided views about the winners and losers of the platform and handset race for 2011, and discussed the challenges facing mobile network operators in their quest to stay relevant to mobile application developers [8].

On the Software Engineering side, in 2010, Tony Wasserman analyzed the issues for mobile development, and provided a view about mobile development environments [9]; moreover, Jeff Rowberg proposed a comparison between different approaches to, and technologies for, mobile development [10].

Subsequently, a comparison between native, web, and hybrid mobile app development approaches was provided by WorkLight Webinar Series [7].

Many other previous studies are concerned with specific platforms, platforms vs. technologies, and development approaches, including: HTML5 versus Android: apps or web [11], Dojo 1.6 [12], PhoneGap to build native apps [13], Appcelerator vs. PhoneGap vs. Adobe Air [14], PhoneGap vs. Flex vs. Appcelerator vs. Corona [15].

Some other studies are concerned with how to correctly exploit technologies, e.g. how to use PhoneGap and the Dojo Toolkit for developing hybrid mobile applications [14].

This paper tried, in its turn, to put together competencies from academic research and a world major company for doing a step ahead in the support of decision making in the mobile apps domain.

11 Conclusions and Future Work

This paper presented a model and guidelines for supporting the choice of the right technological stack for the development of a new mobile application from given

requirements. The proposed model takes into account non-functional requirements, app features, and available development technologies, platforms and approaches. The model is not exhaustive with respect to the technology dimensions and needs; however, it is very simple to utilize, augment and improve by adding new items and attributes. The twelve needs taken into account in this paper are the ones we deem mandatory to consider when making decisions, but their enrichment could boost both quality and type of results.

The method has been empirically tested and further developed by four case studies at the IBM Italy Rome Smart Solutions Lab. These case studies were concerned with different problems to address and covered a wide range of possibilities. However, the number and types of case studies that we utilized to gain experience and eventually populate our base of knowledge (i.e., a simple table) are still limited. Consequently, the items presented in Ordinal scale by Table 1 should not be intended as conclusive values.

Next steps of this study include the development of a larger mix of case-studies, and the study of dependences, if any, of the model on the application domain. One more study we aim to conduct is a deep sensitivity analysis to assess the model robustness with respect to different features and inputs.

Moreover, an extension of the model is planned, which takes in account the value and risk of technologies and processes, as well as the in-house skills and experience.

Furthermore, a theoretical investigation aimed to fully formalize our multi-dimensional space and its mathematical foundation is in our plan for the future.

Last but not least, we aim to create a web site, to be available to the community, where practitioners can share their usage experiences with our framework. This way, it will be possible to gather relevant information from a high number of cases, contexts, and developers. Such information will be used to improve the proposed model and could lead to a new proposal of development process in the domain of the mobile apps.

References

1. Keeney, R.L., Raiffa, H.: Decision with Multiple Objectives: Preferences and Value Tradeoffs. University of Cambridge (1976)
2. Falessi, D., Cantone, G., Kazman, R., Krutchen: Decision-making Techniques for Software Architecture Design: A Comparative Survey. ACM Computing Surveys 43(4) (2011)
3. Zelkowitz, M.V., Wallace, D.R.: Experimental Models for Validating Technology. IEEE Computer 31(5) (1998)
4. Basili, V.R., Weiss, D.: A methodology for collecting valid software engineering data. IEEE Transactions on Software Engineering 10(6) (1984)
5. Wohlin, C., Runeson, P., Horst, M., Ohlson, M.C., Regnel, B., Wesslen, A.: Experimentation in Software Engineering. An introduction. Kluwer Academic Publishers (2000)
6. Fling, B.: Mobile Design and Development: Practical Concepts and Techniques for Creating Mobile Sites and Web Apps, pp. 13–27. O'Reilly Media (2009) ISBN 978-0-596-15544-5
7. WorkLight Webinar Series: Native Web or Hybrid Mobile App Development (2011), `http://www.worklight.com/assets/files/Native-Web-Hybrid-Mobile-App-Dev-Webinar.pdf` (accessed on September 10, 2011)

8. VisionMobile: Developer Economics 2010 and Beyonds (2010), http://www.visionmobile.com/blog/2010/07/developer-economics-2010-the-role-of-networks-in-a-developer-world/ (last access August 12, 2011)
9. Wasserman, A.I.: Software engineering issues for mobile application development. In: ACM SIGSOFT FoSER (2010), http://www.cmu.edu/silicon-valley/wmse/wasserman-foser2010.pdf
10. Rowberg, J.: Comparison: App Inventor, DroidDraw, Rhomobile, PhoneGap, Appcelerator, WebView, and AML (2010), http://www.amlcode.com/2010/07/16/comparison-appinventor-rhomobile-phonegap-appcelerator-webview-and-aml (accessed on January 03, 2012)
11. Google I/O: HTML5 versus Android: Apps or Web for Mobile Development? (2011), http://www.google.com/events/io/2011/sessions/html5-versus-android-apps-or-web-for-mobile-development.html (last access on November 15, 2011), http://www.youtube.com/watch?v=4f2Zky_YyyQ (last access August 14, 2012)
12. Lennon, J.: Get started with Dojo Mobile 1.6 - And get a peek at new features coming in 1.7, IBM developerWorks (2011), http://www.ibm.com/developerworks/web/library/wa-dojomobile/index.html (last access December 13, 2011)
13. Yao, M.: What are the pros and cons of using PhoneGap to build native apps? (2011), http://www.quora.com/What-are-the-pros-and-cons-of-using-PhoneGap-to-build-native-apps (accessed on January 03, 2012)
14. Lukasavage, T.: Review: Appcelerator vs. PhoneGap vs. Adobe Air (2011), http://savagelook.com/blog/portfolio/appcelerator-vs-phonegap-vs-adobe-air (accessed on December 28, 2011)
15. Vilches, A.: PhoneGap vs. Flex vs. Appcelerator vs. Corona: nuevas conclusiones (2011), http://www.xydo.com/toolbar/27165658-phonegap_vs_flex_vs_appcelerator_vs_corona_nuevas_conclusiones_%C2%AB_yo_programador (accessed on January 03, 2012)
16. Dingsor, A.: Writing a Hybrid Mobile Apllication with PhoneGap and the Dojo Toolkit. IBM (2011), http://public.dhe.ibm.com/software/dw/web2mobile/07072011_dingsor/hellohybrid.pdf (accessed on January 04, 2012)
17. Cantone, G., Donzelli, P.: Software Measurements: from Concepts to Production, T.R. Intl. Software Engineering research Network, ISERN T.R. 97-27 (1997) (In Italian)
18. Cantone, G., Donzelli, P., Pesce, G.: Misure software: teoria, modelli e ciclo di vita, in Metriche per il. Software, Ed. GUFPI-ISMA, Franco Angeli (2006) (In Italian)
19. Mastrofini, M., Cantone, G., Shull, F., Diep, M., Seaman, C., Falessi, D.: Enhancing the System Development Process Performance: a Value-Based Approach. In: Procs. of INCOSE 2012, Rome, Italy (2012)
20. Kirby, R.C., Siebenmann, L.C.: Foundational Essays on Topological Manifolds. Smoothings, and Triangulations. Princeton University Press (1977)
21. Boehm, B.W.: Value-based software engineering: Overview and agenda. Tech report, USC-CSE-2005-504, University of Southern California, Park Campus. Los Angeles, CA, USA (2005)
22. Basili, V.R.: Quantitative evaluation of software engineering methodology. In: Proc. First Pan Pacific Computer Conf., Melbourne, Australia, September 10-13 (1985)
23. Cantone, G., Donzelli, P.: Production and Maintenance of Software Measurement Models. Journal of Software Engineering and Knowledge Engineering 5 (1998)

Table 1. The graphical view of an instance of our model for some actual needs, device features, tecnologies, and platforms

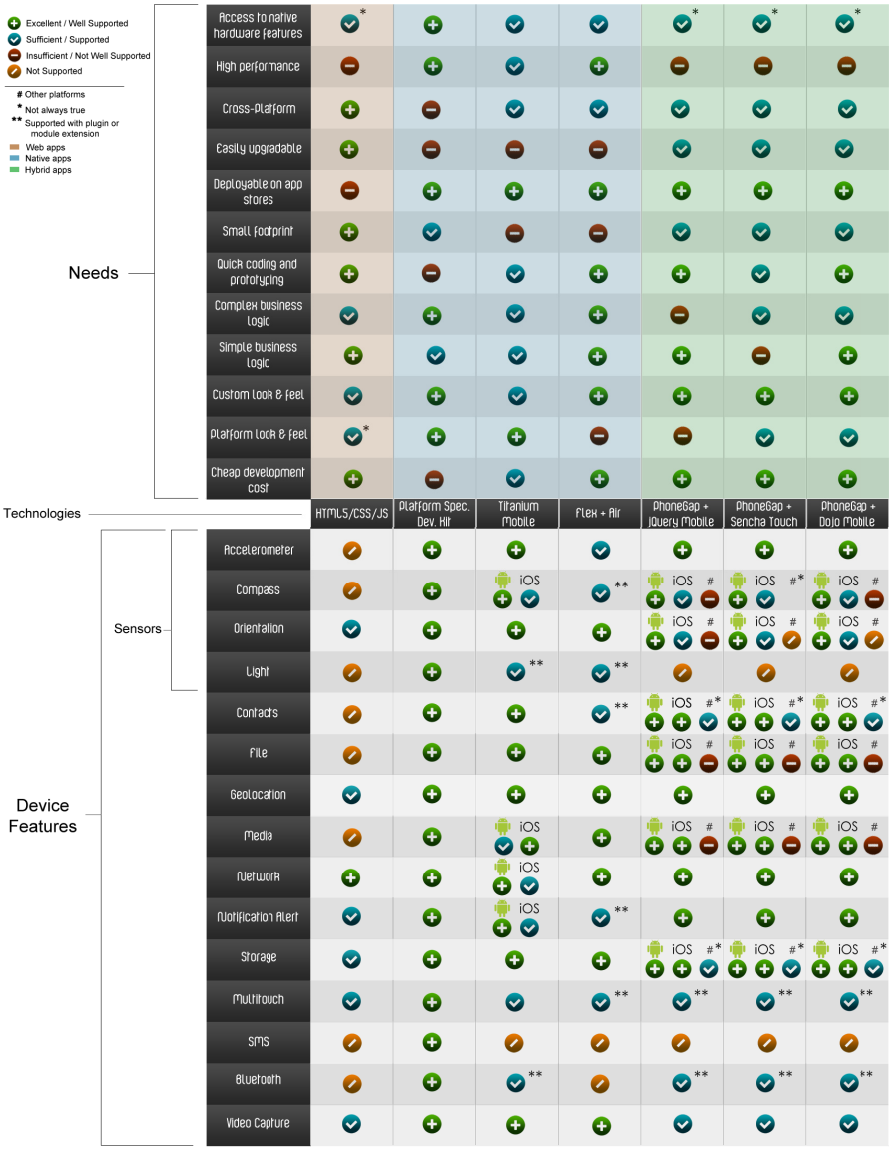
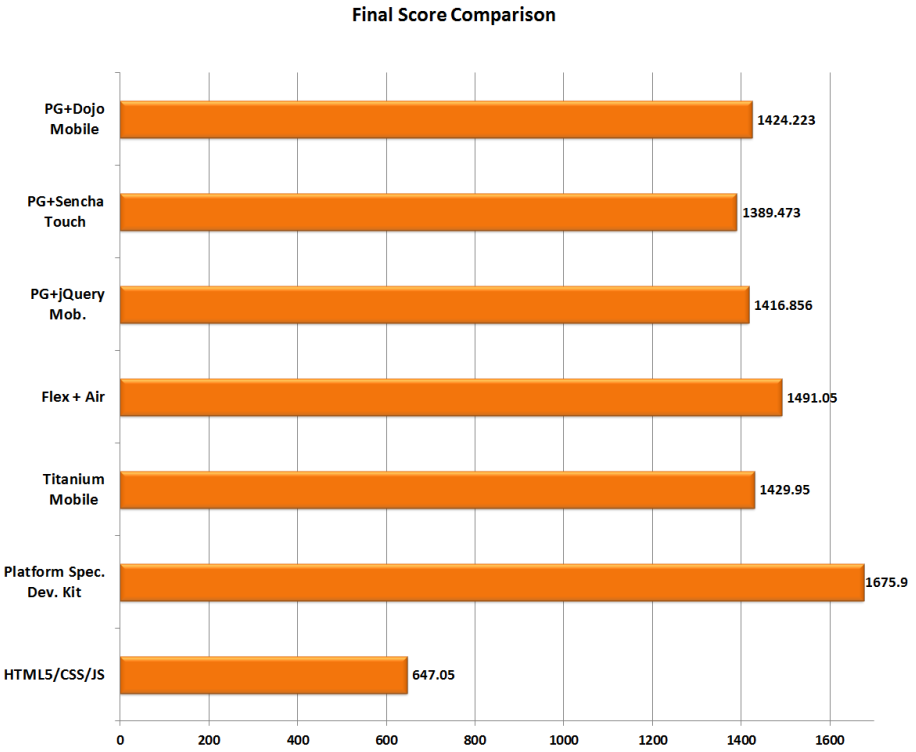


Table 2. Requirements of a Case-study (simplified version)

Description	The app is required to allow an user to chat with other people connected to network by the same application. Moreover, the app is required to provide the user with the ability of filing and sending recorded audio clips and video messages. Furthermore, the app is required to allow an user to extend her/his contact list by specifying the telephone number of the person s/he wants to include; her/his mobile phone list of contacts is expected to be allowed as a source for such a phone number.	
Requirements	1. Available on Android and iOS 2. Access to media 3. Access to videocamera 4. Access to local files 5. Access to contacts list	6. Notification Alert 7. High performance 8. Deployable on app stores 9. Cheap development costs 10. Access to network

Table 3. Results from the Case-study



"TactiGlove" – A Guidance System to Effectively Find Hidden Spots in 3D Space

Georg Stevenson, Andreas Riener, and Alois Ferscha

Johannes Kepler University Linz, Institute for Pervasive Computing
A-4040 Linz, Austria
lastname@pervasive.jku.at

Abstract. Research on vibrotactile navigation and guidance has been a topic of broad interest for the HCI community for quite a while. However, common to most of the presented approaches is a 'map-like' navigation to reach waypoints or targets in two-dimensional contexts. Motivated by the need to simplify and speed up searching tasks in maintenance, training or other everyday situations, a prototype for a three-dimensional guidance system, composed of a glove with integrated 6-DOF positioning sensor technology and vibrotactile actuators, was developed. In opposition to the formerly mentioned systems, the purpose of our "TactiGlove" system is to guide persons to point-shaped, maybe hidden locations in 3D space or annotated digital objects that have been lost. In this paper we present the implemented system in detail, focusing on hardware and software aspects as well as on the different developed tactile feedback models. System evaluation with a lab-based user study based on a Fitt's law experiment for traditional UIs revealed that guidance by the "TactiGlove" was intuitively understood by most of the volunteers, although the system performance varied a lot between the different feedback models. Preliminary results identified much room for improvements and motivates to pursue research on the "TactiGlove".

Keywords: Tactile glove, 3D guidance system, Augmented reality, Improving maintenance/training tasks.

1 Tactile Guidance in 3D Environments

Presenting information using the tactile channel has been an important research topic in the last decade, motivated by reasons such as the possibility to augment or even fully replace information formerly presented using visual and/or auditory channels. This approach would be in particular beneficiary in situations where these senses are either blocked (e. g., in dark or smoky environments where the human eye cannot gather information) or overloaded (for example when maintaining an unknown machine with lots of controls, where the user does not know "where to look" at).

Navigation focusing on tactile feedback only is a well researched topic in this context; the work of van Erp *et al.* is a promising example of such as system developed to support vehicle operators in steering their cars by presenting them

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 80–99, 2013.

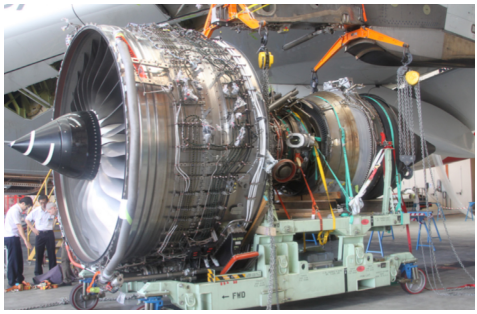


Fig. 1. Tactile guidance could help in improving complex repairing tasks by 'pointing' to the part to change/repair (Image courtesy Australian Transport Safety Bureau, www.atsb.gov.au)

navigational information through the tactile channel [1,2,3]. This enables drivers to keep their eyes and attention on the road and, in addition, to react on recommendations from the navigation system received via tactile messages. Another domain of vibrotactile guidance has emerged in computer aided surgery (CAS). The work of Brell and colleagues [4,5,6] shows impressively how vibrotactile actuators on the hand can guide a surgeon to the area of surgery, even if the patient is thousands of miles away.

The goal of this work is to provide accurate vibrotactile guidance in three-dimensional contexts (such as the one shown in Fig. 1). This is –compared to similar related work– novel as it introduces the advantages of 3D guidance on medium scale (i. e., within reach of the hand or within indoor walking distance). Previously, tactile guidance in 3D was only offered in the large, for example in combination with GPS in aircraft navigation, or at very small scale with pen-shaped interfaces in computer aided surgery. More detailed, the work is motivated by the aim to supply guidance for locating digitally annotated, but hidden, lost or invisible objects. Such a system could enhance maintenance or training tasks, for example on unknown machines, by providing guidance to the object to interact with, and consequently reducing the time to search for the target. Also in everyday situations, for instance if someone has to find an unknown light switch in a dark room, such a system could ease locating it. Considering possible application fields we decided to implement the system as glove, as in most situations the hand would be the extremity responsible for interaction with the object to locate. A further motivation for using the palm as vehicle for our purpose stems from the fact that it offers very low two-point discrimination threshold [7], i. e., allows for a small-sized interface.

1.1 Research Approach

We hypothesize that utilizing vibrotactile guidance provide persons assistance in finding arbitrary locations or annotated objects in three-dimensional space.

Furthermore, we suppose that humans are able to understand and use such signals effectively after a short training phase (on usage of carefully/properly designed signal patterns). The term "effectively" is understood and used here as indication that objects (targets) of different size and distance one to the other are all found in similar temporal behavior.

To evaluate the hypotheses we developed a prototype device that was further on tested in a lab-based user study. The development phase included also the definition of appropriate signal patterns and navigation models. A navigation model in this context describes how position and orientation data of a certain tracking system are transformed and mapped to vibrational patterns presented to the user. With regard to the hardware/software setup, the main requirements were to run the system in real time, implying that both a tracking system with high update rates and fast responding tactile actuators have to be used. Beside the core components, also a reliable wireless network link was required to interconnect the different components without significant delays. For evaluation purpose volunteers had to follow a "virtual" (invisible) route, composed out of predefined, counterbalanced way points of variable size and distance. The measured performance was then evaluated with a metric similar to "Fitt's law" [8] – the standard for traditional interface performance evaluation.

1.2 Outline

The rest of the paper is structured as follows. Section 2 gives an overview of related work and state of the art in vibrotactile guidance systems. Section 3 presents the developed system, section 4 gives insight into the user study and the results of the evaluation. The final section 5 summarizes the findings and achievements and concludes the paper with recommendations for improvements.

2 Related Work

Bosman and others [9] proposed a pedestrian guidance system called GentleGuide. It consists of two vibration units mounted on to the users wrist and provided signals whether a user should turn left, right, moves into the wrong direction, or reached the target. In contrary to our system only map-like 2D guidance to reach waypoints inside a room was provided. Van Erp and colleagues introduced a tactile navigation system mounted on a belt and used that system for finding waypoints on a map in different driving scenarios [2]. Also in this work the navigation task was limited to 2D space. The summary of the research work on tactile navigation systems [3] describes an example of a three-dimensional navigation system. Contrary to our "TactiGlove" system, their proposed 3D navigation device was designed for navigation in a vehicle/aircraft context. The aim of the system was to find outdoor targets using GPS positioning and not to find annotated small objects in an indoor environment. Hein and Brell [4] proposed a tactile guidance system to be used in computer aided surgery (CAS). The system assists a surgeon by navigating his hand, along with the surgical instrument, to the area of interest where it should be used. The tactors 'tell' the surgeon in which direction to

Table 1. The "TactiGlove" is the first tactile 3D guidance system for the hand

System	Integration	Body location			Dimension		Substitution level		Target size		
		Torso	Hand	Wrist	2D	3D	Support	Substitution	Small	Medium	Large
GentleGuide [9]	wristband			X	X			X		X	
Ground-based Waypoint Nav. [2]	belt	X			X			X			X
Helicopter Waypoint Navigation [3]	belt	X				X		X			X
conTACT [4]	glove		X		X		X		X		
Tactile Wayfinder [10]	belt	X			X			X		X	
AR Target finding [11]	cell phone		X		X		X			X	
LifeBelt [12]	belt	X			X		X			X	
CAS Belt [13]	belt	X			X		X		X		
Passive Music Learning [14]	glove		X				X		X		
"TactiGlove" (proposed system)	glove		X			X		X	X	X	

move on the patients surface – this is also a two-dimensional guidance approach (although the system gathers and process three-dimensional data sets). Heuten and others [10] propose a system called tactile wayfinder, which consists of a belt with tactors and a GPS receiver. The difference to the system of Van Erp *et al.* described before is the tactile wayfinder targets pedestrian navigation, thus navigation on much smaller scale (but still outside and in the range of tens to hundreds meters). Ahmaniemi and Lantz [11] proposed to use current Smartphones (all equipped with sensors and actuators) for vibrotactile guidance. They implemented sort of point of interest (POI) navigation by setting off vibration units when the user points at the target (with the phone in the hand). Although no real guidance is provided (the device 'displays' only if the user points in the right direction), it is also said to be a kind of two-dimensional (guidance) system. Ferscha and Zia [12] proposed the 'LifeBelt' - a belt type tactile guidance system similar to the ones of Heuten [10] or van Erp [3], but with more advanced guidance information. They actually incorporate walking trajectories of persons nearby and calculate the best (i. e., most efficient) way to reach a certain target in crowded environments. Another waistbelt system used in a CAS context was presented by Bluteau and others [13]. It signals on a two-dimensional plane whether or not the surgical instrument is moved on the correct trajectory. Huang, Starner and others [14] presented a vibrotactile assistance system integrated into a glove and employed to teach piano scholars to play piano melodies. Vibrators are placed on top of each finger in the glove; activated tactors indicate the finger to be used for playing the next note. While not describing a "guidance system" this device is one example of success for tactile information transmission on the fingers/hand. A summary of the related work with emphasized characteristics and unique features is presented in Table 1. (With regard to the feature 'target size', small stands for targets sized one meter or less, medium targets are 1 to 5 metes large, and big indicated targets larger than five meters.) This overview clearly points

out that "TactiGlove" is the first 3D indoor navigation device for the hand. Worth to mention here is the ARMAR (Augmented Reality for Maintenance and Repair) project [15]; although it does not use tactile patterns to provide directional information it features a similar idea as the one used by "TactiGlove" and employs it also for maintenance purposes. ARMAR uses a visual augmented reality overlay of the real world to guide mechanician to find the next task and action to perform.

3 Prototype Development

3.1 System Architecture

The prototype is technically a sensor-actuator system that transforms input data into several output streams. Input comes from a tracking system that measures the position and orientation of the tactile glove. This input data is then processed in the software engine of our prototype to create precise and accurate vibrotactile output for a certain navigation model. In addition, a visualization of the measured situation (i. e., glove movement in the 3D room) is generated. As common 3D engines normally used for scientific simulations, gaming, or complex visualizations have a similar "update and presentation" architecture, and because these engines already feature built-in vector/matrix processing and visualization libraries to efficiently handle 3D data, we selected such an engine (the Microsoft XNA framework) as base for our software implementation and

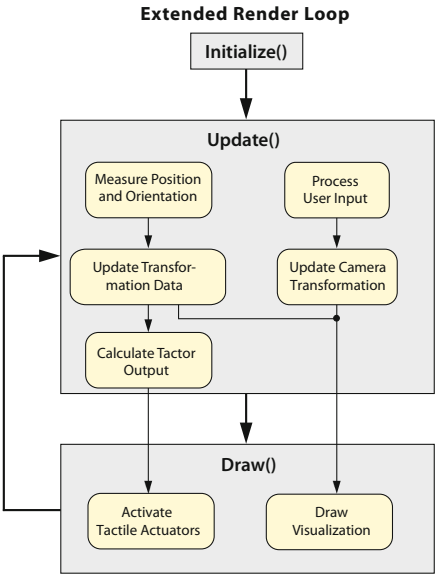


Fig. 2. Render loop with sensor measurements and tactor output

built our system around it. 3D engines normally featuring also real time operation; by extending such an engine the real time requirement of our 3D tactile guidance system was also achieved. The render loop (Fig. 2 shows the control flow) of our 3D engine is structured as follows. First `Initialize()` loads all the necessary data and uploads it to graphics memory. In our case, additionally to initialize the visualization part, all the connections to the tracking system and the controller of the tactile actuators are established in this phase. Then the actual render loop starts with a cyclic repetition of calls of `Update()` and `Draw()` routines. Typically, in `Update()` control signals (e. g., button presses, mouse movements, or time based updates such as animations) are processed, which update the transformations and states of the 3D objects. In our prototype the input is driven by processing data of the tracking system. Updating the internal models also means calculating the current state of our navigation models. In the `Draw()` method the uploaded vertices are transformed using a set of transformation matrices, then rasterized, and finally presented on the screen. Concurrently, the control signals to drive the tactile actuators are forwarded to the tactor control unit (according to the actual navigation model).

3.2 Hardware and Software Setup

To track the position of the "TactiGlove" the Intersense IS-900 motion tracking system was used. This high-precise measurement system is operating based on ultrasonic and inertial signals to provide tracking data with an accuracy of about $2mm$ (position) and with $0.50°$ rotational resolution (pitch, roll, yaw) at update rates of up to $180Hz$. These specifications as well as the update rate are by far sufficient for our "TactiGlove" system as we wanted to achieve an update rate of about $30fps$ and a "guidance resolution" of at least $25mm$ (i. e., to guide to small objects sized approximately $5cm$). For generating the vibrotactile output C-2 tactors from EAI technologies were used. These tactile actuators are small enough ($30mm$ diameter, $7.9mm$ height) for being integrated into a common glove, and featuring strong, continuous, and clearly recognizable vibration signals at an update rate higher than $30Hz$. The IS-900 system was directly connected to the host computer using USB and the tactor commands were transmitted from the host computer to the tactor control unit (TCU) using a Bluetooth connection. Both the Intersense tracker and the tactor elements were integrated into a glove worn by the user (="TactiGlove"). In order to butt the tactors against the palm skin a tight fit glove typically used by bicyclers was selected as "tactor housing". Tactors were fixed to their defined positions with elastic bands/straps that were sewn on the inside of the glove. The tracker was mounted on the back of the hand using stitched elastic bands. Its position was chosen in a way that the tracker has "good" visual connection to the "SoniStrip" units responsible for receiving and processing ultrasonic wave signals from the tracker all the time. The integration of all the components into the "TactiGlove" prototype is shown in Fig. 3. The software implementation is based on Microsofts .NET 4.0 and XNA frameworks. It integrates the DLLs for interfacing with the IS-900 system and the tactor controller and was developed in a way such that

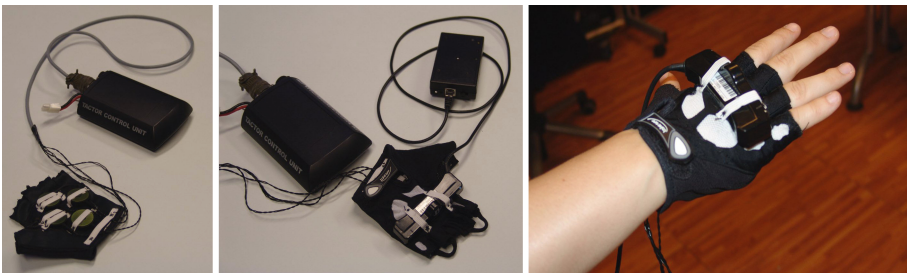


Fig. 3. Four tactors mounted on the palm side of the glove (first row, left); upper side of the glove with the Intersense tracker (first row, right); "TactiGlove" prototype worn by a user and with all the components integrated (second row)

not only signal processing and tactor control is handled by the software, but that also use cases/scenarios for the user study can be configured, monitored, recorded, visualized, and evaluated within the self-contained environment.

3.3 Navigation Models

Navigation models describe how the data received by the tracking system is processed and transformed to tactile stimuli. In this work, a navigation model describes (i) where on the hand the tactors are placed, (ii) which vibration patterns are used to activate them, and (iii) which meaning such a '*Tactogram*' [16] assigned to a specific tactor has. It further includes the algorithm to determine which vibration pattern has to be assigned to the individual tactors on the glove. In our work, Tactograms are only distinguished by variation of the two parameters vibration intensity and frequency. As later shown in the evaluation section, three navigation models were used to study the efficiency and intuitivity of vibrotactile guidance in 3D space. Common to all the 3 models is the need for transformation. Tactor positions are stored as three-component vectors in a coordinate system relative to the origin of a virtual hand model (in computer graphics often referred to as the 'model space'). From the IS-900 tracking system the position and orientation of the tracker is retrieved in its own coordinate system. Its origin is also taken as origin of XNAs world coordinate system, where every model in model space is transformed to in order to form the entire scene. To recapitulate, tactor positions in the model space have to be transformed into world coordinates so that they represent their position in the real space. For that, transformation matrices (for translation and orientation) are generated from the position/orientation data of trackers. The final world space position of a tactor can then be obtained by calculating

$$P_{world} = T \cdot R(yaw, pitch, roll) \cdot P_{model}$$

where P_{world} is the tactors position in world space, T is the translation matrix formed by the position data from the tracking system, $R(yaw, pitch, roll)$ is the

rotation matrix formed by the orientation data in Euler angles from the tracking system, and P_{model} is the tactors position in model space. Details on how these matrices are processed can be found, for example, in [17].

'Plane Model'. This model orients itself on the frequently used belt-type navigation systems as indicated for instance in [1,9,10,12]. Four tactors are used to indicate in which direction the target is located. If the target is in the direction the hand of the user is actually pointing to than the front tactor is vibrating, if the target is in opposite direction the back tactor vibrates. The same approach is followed for targets placed left, right or in diagonal directions. The classification of which tactor has to vibrate is done by determining in which angular sector relative to the hand the tactor is located (Fig. 4).

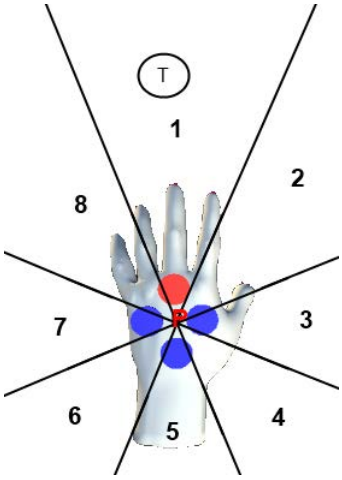


Fig. 4. Plane Model: Fragmentation into sectors to classify the active tactors. If a target 'T' is located within a certain sector the according tactor(s) vibrates. In "TactiGlove" we use eight sectors mapped to four tactors. For the diagonal sectors (2, 4, 6, and 8) the two adjacent tactors are activated. Red tactors are on, blue colored tactors are switched off.

To extend this behavior to the third dimension we used two mechanisms. First, the tactors can indicate if the target is located approximately orthogonal to a plane described by position and orientation of the Intersense tracker. Approximately means in our case if the angle between the normal of our simulated plane and the target is below 45°, and therefore describing a cone around the normal. If the target is located inside the angles described by the cone than all four tactors are activated to signal the "TactiGlove" bearer that the target is orthogonal to its hand. Fig. 5 illustrates this behavior. The second mechanism to incorporate the third dimension simply takes the position and orientation data of the tracker to transform the plane accordingly to the users hand. If the user finds that, for example, the target is above or below the users hand, it is possible

to simply tilt the hand so that the target is no longer orthogonal to it, therefore again more detailed direction cues are presented (in the tilted plane). Before calculating the directions (in the case that the target is not inside the sector described by the orthogonal cones) the target has to be projected orthogonally on the virtual plane. This ensures that the vector between the position of the hand and the projected target can be used to compute the directions. To project the target to the virtual plane the plane equation $v_n * X = v_n * P_h$ was used, with v_n corresponding to the normal vector of the plane and P_h representing the position of the hand. This was then combined with the line equation $X = P_t + t * v_n$ that describes a line orthogonal to the plane at the position of target P_t. After inserting the line equation into the plane equation, t can be calculated as shown in the equation $t = (v_n * P_h - v_n * P_t)/(v_n * v_n)$. To calculate the angles necessary for the behavior above we used four vectors v_1 to v_4 assigned to each tactor, each pointing to the direction assigned to the tactors in untransformed model space, therefore $v_f = (0, 0, 1)$, $v_r = (1, 0, 0)$, $v_b = (0, 0, -1)$, and $v_l = (-1, 0, 0)$. In addition, one vector is assigned to the orthogonal plane $v_n = (0, 1, 0)$. These vectors are then transformed using the transformation described before. As next step, the vector $v_{target} = P_t - P_h$ is calculated, with target position P_t and hand position P_h. Using the law of cosines the angle between the five transformed vectors and v_{target} is computed. $\alpha = \arccos((b^2 + c^2 - a^2)/(2bc))$ where $b = P_t - P_h$, $c = (P_h + v) - P_h$ and $a = (P_h + v) - P_t$. To calculate the directional angles, the projected target position is used instead of the real position. After the angles were calculated they are simply compared to threshold angles which specify if the according tactor shall be activated or not. For the directional vectors $v_f v_r v_b v_l$ this angle was set to 67.5°, causing the sectors to overlap so that the overlapping

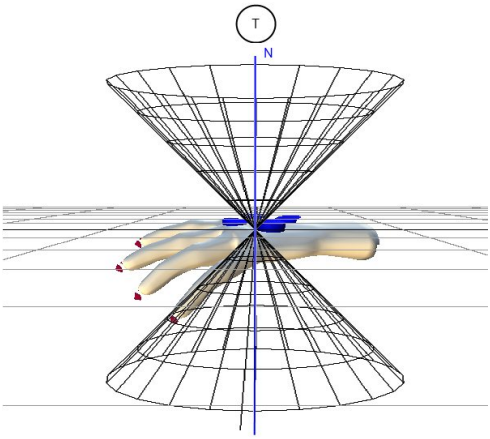


Fig. 5. Plane Model (system extension to 3D space): A target 'T' located inside the cone (as shown in the figure) causes all four tactors to vibrate. This signals the "TactiGlove" user that the target is approximately orthogonal (i. e., above or below) to the plane described by the hand of the subject. A target outside the area described by the cones causes tactor activation according to the description in Fig. 4.

areas represent exactly the diagonal sectors. For the orthogonal vector v_n the angle was set to 45°.

'Yaw-Pitch Model'. The second model orients itself on the idea presented in [11]. Here accelerometers were used to determine the pointing direction of a user, and based on this information a tactor informs the user whether or not the actual pointing direction accords with the direction of the target. In [11] this was only realized for yaw angles, which means that only two-dimensional scenarios were covered. We extended this model by including a second tactor presenting tactile information regarding the pitch direction, i. e., indicating if the user also points towards the target with correct pitch angle. By including up/down directions a "TactiGlove" bearer is now enabled to look for and find targets in three-dimensional space.

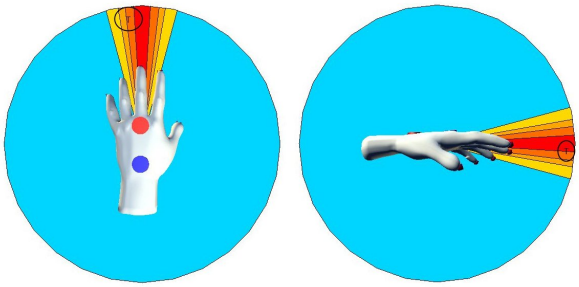


Fig. 6. Yaw-Pitch Model: Angular sectors with color coded vibration strength of the tactors (relative to the yaw and pitch angles by the hand of the bearer). Red color (facing exactly to the front) indicates maximum vibration strengths, blue color stands for no vibration at all. The color grading between red and blue encodes vibration intensities between maximum strength and off. In the left figure, target 'T' causes the front tactor (mapped to yaw angle) to vibrate at second highest gain level, while the back tactor (right figure; pitch angles) vibrates at full strength. In the real system, both yaw and pitch vibrations are overlayed and indicated at the same time.

In the 'yaw-pitch model' we are using different signal strengths to signal if a subject is pointing only roughly or exactly in the right direction. To compute the strengths of the signal the angles between the vector in viewing direction v_{view} and the vector between target and hand v_{target} projected in XZ and YZ planes have to be calculated in order to classify in which angular sector the target falls. The sectors are mapped then to categories of vibration signal strengths, from no vibration (blue area in Fig. 6) to full vibration if the "TactiGlove" bearer exactly points towards the target (red sector). For the mapping between signal strength and pointing angle a Gaussian distribution was used such that the angular sectors are getting more narrow the more precise the user is pointing toward the target. The threshold angles to choose the tactor strength were chosen as follows. If the according angle is below 3.5° the tactor vibrates at full gain, between 3.5° and 7.0° at medium-strong gain, between 7.0° and 10.0° at medium gain,

and between 10.0° and 15.0° at weak gain. An angle higher than 15.0° causes the tactor not to vibrate at all. Fig. 6 illustrates the angular sectors with color coded vibration strengths. For calculation of the yaw angles in between we simply project v_{target} and v_{view} against the XZ plane by skipping the Y component. Therefore $v_{targetXZ} = (x_{target}, 0, z_{target})$ and $v_{viewXZ} = (x_{view}, 0, z_{view})$, where v_{view} is the transformed direction of the hand and $v_{target} = P_t - P_h$. The viewing direction is calculated from a vector that is by default $(0, 0, 1)$ and is then transformed by the matrices constructed from the data received by the Intersense position/orientation tracker with P_t being the position of the target and P_h the position of the hand. The angle Δyaw between v_{viewXZ} and $v_{targetXZ}$ is calculated as angle between two vectors $\Delta yaw = \arccos \frac{v_{viewXZ} * v_{targetXZ}}{|v_{viewXZ}| * |v_{targetXZ}|}$. The pitch angle difference is calculated as the (absolute) difference between the height angle of the target relative to the hands position and the given pitch angle from the tracking system. The height angle α is calculated from the law of sines $\alpha = \arcsin \frac{b * \sin 90}{|v_{target}|}$ where $b = P_t.Y - P_h.Y$. α is later referred to as Δ_{pitch}. Δ_{pitch} is taken as $\Delta_{pitch} = 360° - \Delta_{pitch}$ if the angle was above 180°. This case can occur if the Intersense tracker faces downward and therefore the palm of the user is facing, by mistake, upward.

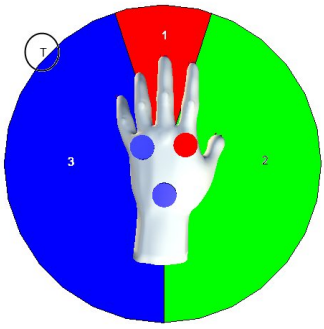


Fig. 7. Yaw-Altitude Model (top view) with angular sectors of yaw angle. If a target is residing in sector 1 (red) both "angular tactors" vibrates, if it is in sector 2 (green) only the left tactor vibrates, if it is in sector 3 (blue) the right tactor vibrates. Target 'T' in the figure causes the right tactor to vibrate, signaling the "TactiGlove" bearer to turn to the right. This is sort of inverse model and the same paradigm is followed for the altitude (one tactor mounted above, one below the hand).

'Yaw-Altitude Model'. The third model mixes cues of pointing and instruction directions and can be somehow seen as "inverse model". It 'tells' the "TactiGlove" bearer if he/she is pointing in the right yaw direction and furthermore if the height position of the hand is in the right altitude or if the hand has to be moved upwards or downwards. In contrast to the 'yaw-pitch model' above, two tactile actuators instead of only one tactor are used to present the yaw angle and/or the altitude of the hand. This means that the "TactiGlove" bearer is not only informed if he/she is pointing in the right direction, but also

if he/she should turn left or right. The same paradigm is used for indicating whether or not the plane of the hand is in correct altitude to reach the target.

Calculation of Δ_{yaw} is done analogous to the 'yaw-pitch model' with the exception that the angle is signed whether the user has to turn left or right or not. To check the turning direction $v_{targetXZ}$ and v_{viewXZ} are used. v_{viewXZ} is then transformed using a rotation matrix given from the determined yaw angle to $v_{viewXZangle}$. If $|v_{viewXZangle} - v_{targetXZ}| = 0$ then $\Delta_{yaw} = \Delta_{yaw}$, otherwise $\Delta_{yaw} = -\Delta_{yaw}$. If the angle between view $v_{viewXZangle}$ and $v_{targetXZ}$ is negative then rotating v_{viewXZ} using Δ_{yaw} exactly matches $v_{targetXZ}$, and therefore the vector in between has a length of zero; in any other case there is vector in between. If the calculated $|\Delta_{yaw}|$ is smaller than 18° (10% of 180°) then both angle tactors vibrate, signaling the user that he points in the right direction. If Δ_{yaw} has a negative sign then the user has to turn right until the direction matches, if Δ_{yaw} is positive, than the user has to turn left. Fig. 7 illustrates this approach.

Altitude signals are calculated straight forward. A target 'T' has additionally to its position a top and bottom border, which can be computed easily from the targets spheres radius. We then have only to check if the Y coordinate of P_{hand} rests below the bottom of the target, above the top of the target, or in between (=correct altitude; no vibration).

4 Evaluation

To evaluate our "TactiGlove" prototype a lab-based user study was conducted; this section presents the procedure and summarizes the results obtained.

4.1 Experiments

To evaluate the behavior of the system all test subjects (see below for details) had to perform movements from predefined starting points to sphere shaped virtual targets. Both (intermediate) start position and target points (objects) were not visible in reality, but were represented as position vectors in our evaluation software. This ensures that searching tasks for invisible objects are correctly simulated. The first starting position was visually marked in the laboratory room, after reaching one of the targets, the user received a short acoustic notification signal. The experiment was designed in a way that the test subjects had to follow a path with explicit direction changes consisting of multiple targets (different distances, different sizes; all parameters counterbalanced). After reaching a target (and receiving notification), this target immediately becomes the starting point for the next search task (see Fig. 8 for explanation). The target spheres had different sizes to represent multiple difficulty levels. Although Fitt's law [8] is not directly applicable as it is normally used to evaluate the performance of movements with input devices (such as keyboard, joystick, or mouse) instead of movements driven by a guidance system, we found it a useful performance metric also to evaluate the performance of the (different models of the) "TactiGlove"

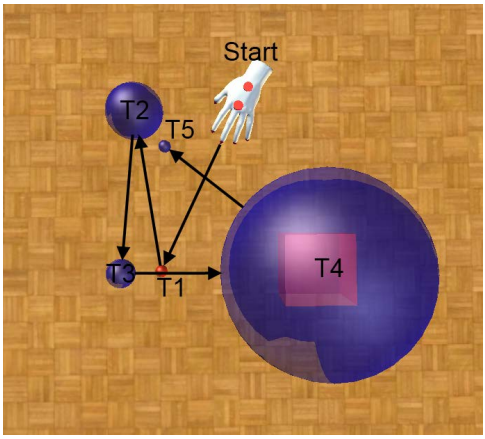


Fig. 8. A path is formed by multiple targets. The first target to reach is T1, then T2, T3, and so on.

system (same variation elements). A participant had to move from a given starting position to a target position, passing a distance or amplitude A. The targets all had different sizes W. This provides us to choose difficulties accordingly to $ID = \log_2(A/W + 1)$ (ID...index of difficulties). We chose A to be fixed at 2 meters, resulting in values of W for each ID as given in Table 2. The fixed distance should not have influenced the behavior of our test subjects, as they haven't been informed about this prior to the study. During initial experimentation it turned out that small spheres with a radius of only about three centimeters (ID 5, 0.0645 m diameter) were very hard to find and took a long time, so they were skipped for the final user study.

One test run had to be accomplished by each test subject per each navigation model, which results in three runs per test subject. One test run contained twelve targets with three instances of each ID. We expected learning effects for each navigation models, but also among them. To compensate, test subjects were given a short training phase per trial where they could get familiar with the navigation model. In a training phase five targets with ID 2 were presented

Table 2. Index of difficulty (ID), target amplitudes (A), and target widths (W) as used in our experiments

ID [bits]	A [m]	W [m]
1	2.00 m	2.00 m
2	2.00 m	0.67 m
3	2.00 m	0.29 m
4	2.00 m	0.13 m
(5)	(2.00 m)	(0.06 m)

and had to be found. A training cycle could be repeated as often the user wants. We counterbalanced the order of navigation models within-subject to cope with learning effects among the different models. There was a time limit of 15 minutes for each trial, but no time limit on a single target. If technical problems occurred or the target was not reachable for the person (e. g., the test subject was too small) the corresponding items were skipped. For each experiment a trace together with temporal data was recorded. These traces allow for visual inspection in order to reason about specifics how the user moved or the time (on a per item basis) can be extracted and used for further evaluation. An example of such a trace is given in Fig. 9. Theoretically, the moved distance could also be derived from such a trace, but jittering and random failures of the tracking system have made these data in the given experiments unreliable.

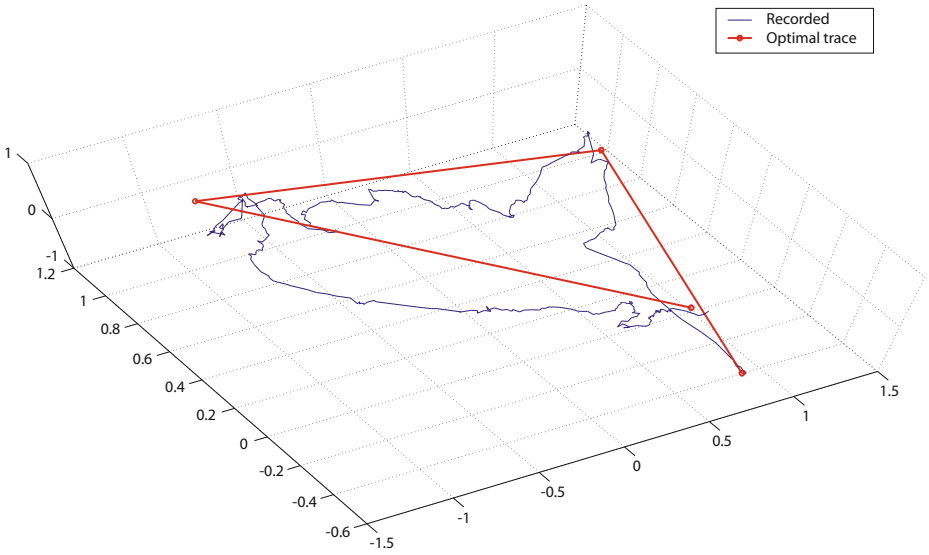


Fig. 9. 3D plot of a recorded trace. The blue line represents a recorded trace, the red line the optimal path between starting point and two targets.

4.2 Results

We tested the prototype system with 17 test subjects (13 males, 4 females; age $\bar{x} = 27 \pm SD = 5.84$). The complete test would have generated a total sample size of 51 measurements for each of the 4 IDs and 3 navigation models (612 samples). However, not all measurements could be used for our evaluation – some of the target traces had to be removed later because of time lapse, while others were skipped due to technical problems with the tracking system (indicated in detail below). An examination of the problems is one aim of this section, another is the performance investigation as well as a comparison for the three different navigation models using a metric similar to Fitt's index of performance (IP).

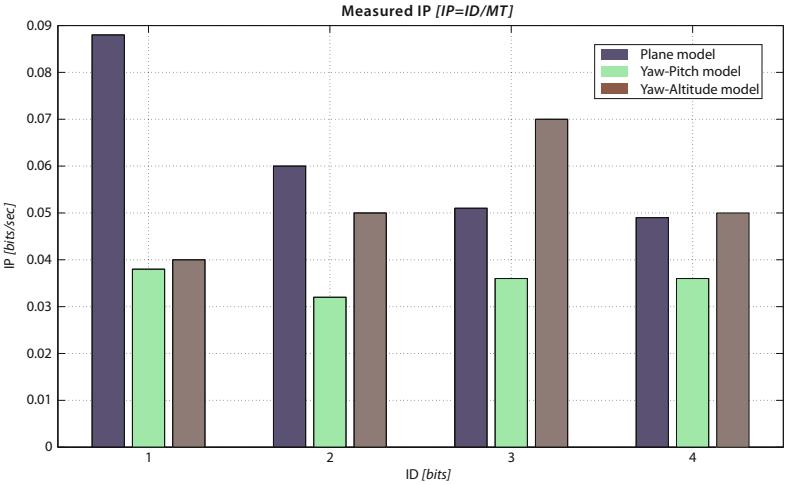


Fig. 10. Performance (IP) for all the three models and different task difficulties (IDs)

In the evaluation for the 'plane model' 16 targets (or 7.8%) had to be removed because of tracking system problems, 18 (8.8%) targets because of time lapse, and additionally 12 (5.9%) due to problems with the glove (slipping/moving during the experiment). These skipped measurements have not been used for the final evaluation presented later. In the remaining 158 correct measurement points only 2 (1.27%) were erroneous, finally resulting in a target completion rate of 98.73%. For the 'yaw-pitch model' 11 measurement points (5.4%) had to be withdrawn due to problems with the tracking system, 32 (11.3%) because of time lapse, and 12 (5.9%) because of glove problems. In the remaining 147 measurements 2 (1.36%) were faulty (target completion rate of 98.64%). In the last series evaluating the performance of the 'yaw-altitude model' 18 targets (8.8%) had to be skipped because of problems with the tracking system, 48 (23.6%) because of time lapse, and 12 (5.9%) due to problems with the glove. These skipped measurements are again not taken into account for the final evaluation. In the remaining 126 measurements only 1 (0.79%) was indicated erroneous, leading to a target completion rate of 99.2%.

It can be observed that most of the skipped targets (about 21% of all measurements) were skipped due to time constraints or problems with the "TactiGlove" (size, slipping during the experiment), another 7% were skipped due to technical problems. This indicates that in scheduling possible future experiments more time should be reserved for each test subject. Nevertheless, the low skip rate caused by technical problems and the high overall completion rate of the actual used measurements let us feel quite positive that the system will operate at ever higher accuracy and without faults in the near future – as it has shown that it works already in its current stage of development very well.

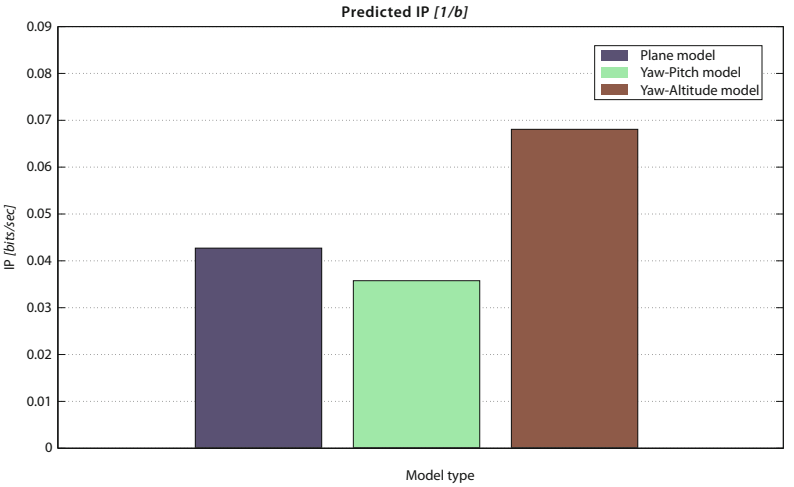


Fig. 11. Predicted performances (IPs) for the three models

To compare the performance of each navigation model against each other the mean values of the measured times for each model and for each ID were taken to build the prediction model $MT = a + b * ID$ (according to Fitt's law [8]). The correlation of measured and predicted MT gives an indication how good the prediction fits the measured data. For the 'plane model' a prediction model of $a = -12.335sec.$, $b = 23.372sec.$ was derived; $MT = -12.335 + 23.372 * ID$, $r^2 = 0.9993$. For the 'yaw-pitch model' the prediction model is specified by $a = 1.215sec.$, $b = 27.826sec.$ and $MT = 1.215 + 27.826 * ID$. The correlation coefficient was calculated at $r^2 = 0.9892$. For the 'yaw-altitude model' finally a prediction model of $a = 1.215sec.$, $b = 27.826sec.$ and $MT = 9.475 + 14.662 * ID$ was derived ($r^2 = 0.8633$). The index of performance per model and ID (Fig. 10) allows the following conclusions. The 'yaw-pitch model' performs worse (it has the lowest IP for all IDs, the 'plane model' performs in our experimental series best for targets with low IDs while the 'yaw-altitude model' performs best for targets with higher IDs.

Finally the performance of a model can be quantified by only one value by taking the reciprocal values of the regression line's slopes into account to derive the value for IP [8, p. 97]. The higher the bit rate the better is the performance of the model. The runs using the 'plane model' achieved an IP of 0.0428, while the 'yaw-pitch model' achieved IP=0.0359 and the 'yaw-altitude model' attained a IP of 0.0682 (=model perceived and performed best in the experiments). The performance of the different models (reciprocal IPs) is shown in Fig. 11.

To complete the evaluation, Fig. 12 shows the regression lines for each model in a single plot to better indicate the differences between the three models used.

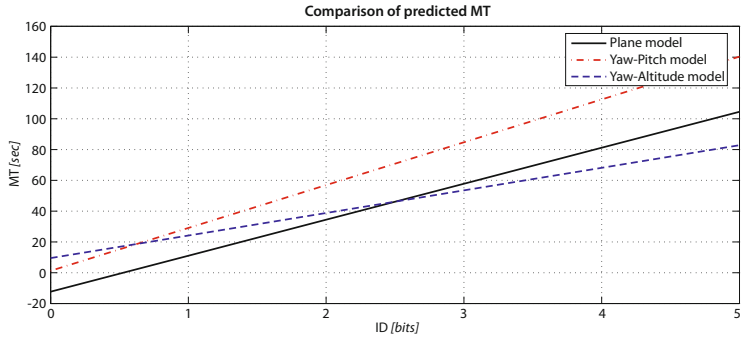


Fig. 12. Predicted movement times (MTs) for the 3 navigation models

4.3 Discussion

The most crucial points to discuss are whether the stated research hypotheses are supported by the executed study and its evaluation. The **first hypothesis** states that **vibrotactile guidance cues allows persons to find arbitrary locations or annotated objects in 3D space** by using vibrotactile signals only. For this hypothesis our test subjects had to complete a series of trials with different difficulties for each model. As shown completion rates of 98.7% for the navigation using the 'plane model' as well as the 'yaw-pitch model' and 99.3% for the 'yaw-altitude model' gives support for this hypothesis. However, a sample size of 17 subjects with each 12 trials per navigation model is fairly small, nevertheless, we tried to diversify the backgrounds of the subjects and used not only people with deep technical knowledge.

Along with the evaluation of the this hypothesis a performance comparison of the proposed model was performed. The comparison revealed that in general the 'yaw-altitude model' achieved the highest bit rate of 0.07 bits/sec. (=index of performance) and therefore worked best. However, the bit rate is quite low compared to other experiments using Fitt's law – mouse input, for example, achieved IP values of up to 4.35 bits/sec. [18]. Nevertheless, the intention of our experiments was not to compare the "TactiGlove" against mouse, touch, trackball, joystick, or other similar input devices, but to give us the possibility for comparing these initial results with prototypes and navigation models developed in the future.

The **second hypothesis** posed was that **test subjects would be able to understand and use such signals effectively after a short training phase** if properly designed signal patterns were used. Every user had a short training phase before executing the actual experiment. The high completion rate could be an indication for the acceptance of this hypothesis, but it does not evaluate how effectively the actually have been used. To answer the question, test participants had to complete a questionnaire after the experiment. Its examination showed that the three navigation models were clear each to everyone; nevertheless, some of the subjects had problems with distinguishing the tactors from each other,

especially after using the system for a while. This is a clear advice that the system has to be further improved in making the tactors better distinguishable and the tactile patterns better perceptible. The majority of the users reported that they have understood the navigation models and could distinguish the signals. Along with the high completion rates the second hypothesis is supported. Many users reported that they had problems with finding the targets, but this more reflects that they had to deeply concentrate on the task. If this problem could be reduced with more training has not been evaluated in the current study. Additionally, we have also shown which one(s) of the proposed navigation models worked best, and which one(s) should be skipped from further investigation.

One weakness of the presented study is that the proposed models were not compared against alternative guidance systems using alternative modalities, such as visual or acoustic guidance. It is expected that looking for known entities using visual search (e. g., by marking a target with an augmented reality overlay) would be much faster than guiding the hand to a target using vibrotactile feedback only; but this is also not directly comparable as the proposed guidance system was designed to augment or replace visual searching. For acoustic guidance different results depending on the situation and scenario could be achieved; this might one issue to include for investigation in studies in the future.

To conclude, the evaluation of the proposed system showed that in principle it works already quite well in terms of locating targets by only using the vibrotactile signal. However, there is still much room for improvements, especially in tactor placement, signal processing, and message encoding. It was shown that subjects had problems with distinguishing the tactors, particularly after the system has been used for a while. The measured times were also quite high, which limits the system to be used only in situations where the visual sense is restricted. Weaknesses can be found in the evaluation itself. The problems here consists of small sample size, no differentiation between age group or gender. After improving the system based on the findings of the current user study, all (at least some) of the identified issues should have been eliminated in future studies.

5 Conclusion

In this paper we presented the "TactiGlove", a prototypic vibrotactile position finding system for 3D environments integrated into a glove. Our main motivation was to guide persons, for example in training or maintenance tasks, to point-shaped maybe hidden spots in 3D space or to provide assistance in finding and locating invisible, digitally annotated objects. System evaluation was performed in a lab-based user study based on a Fitt's law experiment, where subjects had to follow virtual, invisible paths (defined by targets of different size and distances) in 3D space similar to a "click & point" interaction with a common computer mouse. The evaluation revealed a completion rate of 98.7 % for both the 'projection on plane' and the 'yaw pitch pointing' models and 99.3 % for the 'yaw and altitude' navigation model. Furthermore, the guidance by the "TactiGlove" was intuitively understood by most of the volunteers, although

the system performance varied a lot between the different feedback models and subjects. Another finding that might be of help for other researchers is that all the data processing and calculations in 3D space could be easily integrated into a common 3D engine normally used in the gaming, simulation, or virtual reality domains. The three-dimensional data which needs to be processed to drive the "TactiGlove" is very similar to calculations typically carried out in such engines; with our work we have shown that the built-in functions of a 3D engine can be used for efficient sensor data processing and actuator control (e. g., for feedback generation).

5.1 Further Work

Initial experiments with the "TactiGlove" revealed that the prototype basically works well and could be used in the aimed situations, but still offers great potential for further improvements. First, the navigation models, especially the placement of the tactors could be reassessed as many users reported that they were not at all able to distinguish the tactors clearly and really had to put hard effort to concentrate on the task to finally reach the target. From improvements in the placement we also expect increased speed for reaching the targets. Second, the aim of the current study was to test the feasibility of the "TactiGlove". In future experiments the focus should rely more on the reachable performance as compared to other common approaches such as visual search of objects on unknown positions in 3D space or acoustic guidance. Furthermore, future experiments should also feature a bigger sample of subjects with equal distribution of males and females.

Acknowledgments. This work is supported under the FP7 ICT Future Enabling Technologies program of the European Commission under grant agreement No 231288 (SOCIONICAL).

References

1. van Erp, J.B.F.: Tactile Navigation Display. In: Brewster, S., Murray-Smith, R. (eds.) Haptic HCI 2000. LNCS, vol. 2058, pp. 165–173. Springer, Heidelberg (2001)
2. van Erp, J.B.F., Veen, H., Jansen, C., Dobbins, T.: Waypoint navigation with a vibrotactile waist belt. ACM Transactions on Applied Perception (TAP) 2(2), 117 (2005)
3. van Erp, J.B.F.: Tactile Displays for Navigation and Orientation: Perception and Behavior. Ph.D. dissertation, Utrecht University, The Netherlands, Mostert & Van Onderen, Leiden (2007)
4. Hein, A., Brell, M.: conTACT-A Vibrotactile Display for Computer Aided Surgery. In: EuroHaptics Conference, 2007 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics 2007. Second Joint, pp. 531–536. IEEE (2007)
5. Brell, M., Hein, A.: Positioning tasks in multimodal computer-navigated surgery. IEEE Multimedia 14(4), 42–51 (2007)

6. Brell, M., Hein, A.: Tactile guidance in multimodal computer navigated surgery. IEEE Potentials 28(4), 30–35 (2009)
7. Wolfe, J., Kluender, K., Levi, D., Bartoshuk, L., Herz, R., Klatzky, R., Lederman, S.: Sensation & Perception, 2nd edn. Sinauer Associates Inc. (2008) ISBN: 978-0878939565
8. MacKenzie, I.S.: Movement time prediction in human-computer interfaces. In: Readings in Human-computer Interaction, 2nd edn., pp. 483–493. Kaufmann, Los Altos (1995), http://www.yorku.ca/mack/GI92.html (reprint of MacKenzie, 1992)
9. Bosman, S., Groenendaal, B., Findlater, J., Visser, T., Graaf, M., Markopoulos, P.: Gentleguide: An Exploration of Haptic Output for Indoors Pedestrian Guidance. In: Chittaro, L. (ed.) Mobile HCI 2003. LNCS, vol. 2795, pp. 358–362. Springer, Heidelberg (2003)
10. Heuten, W., Henze, N., Boll, S., Pielot, M.: Tactile wayfinder: a non-visual support system for wayfinding. In: Proceedings of the 5th Nordic Conference on Human-computer Interaction: Building Bridges, pp. 172–181. ACM (2008)
11. Ahmaniemi, T., Lantz, V.: Augmented reality target finding based on tactile cues. In: Proceedings of the 2009, International Conference on Multimodal Interfaces, pp. 335–342. ACM (2009)
12. Ferscha, A., Zia, K.: Lifebelt: Silent directional guidance for crowd evacuation. In: International Symposium on Wearable Computers, ISWC 2009, pp. 19–26. IEEE (2009)
13. Bluteau, J., Dubois, M., Coquillart, S., Gentaz, E., Payan, Y.: Vibrotactile guidance for trajectory following in computer aided surgery. In: Conference Proceedings: Annual International Conference of the IEEE Engineering in Medicine and Biology Society, vol. 1, pp. 2085–2088 (2010)
14. Huang, K., Starner, T., Do, E., Weiberg, G., Kohlsdorf, D., Ahlrichs, C., Leibrandt, R.: Mobile music touch: mobile tactile stimulation for passive learning. In: Proceedings of the 28th International Conference on Human Factors in Computing Systems, pp. 791–800. ACM (2010)
15. Henderson, S., Feiner, S.: Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret. In: 8th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2009, pp. 135–144. IEEE (2009)
16. Riener, A.: Sensor-Actuator Supported Implicit Interaction in Driver Assistance Systems, 1st edn., Wiesbaden, Germany, January 14. Vieweg+Teubner Research (2010)
17. Akenine-Möller, T., Haines, E., Hoffman, N.: Real-Time Rendering, 3rd edn. A K Peters, Ltd, Natick (2008) ISBN: 978-1568814247
18. Forlines, C., Wigdor, D., Shen, C., Balakrishnan, R.: Direct-touch vs. mouse input for tabletop displays. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2007), pp. 647–656. ACM Press, New York (2007), http://doi.acm.org/10.1145/1240624.1240726

Towards Multimodal 3D Tabletop Interaction Using Sensor Equipped Mobile Devices

Florian Klompmaker[1], Karsten Nebe[2], and Julien Eschenlohr[1]

[1] University of Paderborn, C-LAB,
Fürstenallee 11, 33647 Paderborn, Germany
florian.klompmaker@c-lab.de, eschenlohr@googlemail.com
[2] Rhine-Waal University of Applied Sciences,
Südstraße 8, 47475 Kamp-Lintfort, Germany
Karsten.Nebe@hochschule-rhein-waal.de

Abstract. Interactive tabletops have been proven to be very suitable setups for collaborative work especially in combination with mobile devices. Further on, many application scenarios require the visualization of 3D data. Therefore we present multimodal 3D interaction techniques for tabletops that allow simultaneous control of six degrees of freedom using sensor equipped mobile devices. In two early user studies we compared multitouch, tangible interaction and sensor equipped smartphones in order to start a User Centered Design process. We got important results regarding effectiveness, intuitiveness and user experience. Most notably we figured out that mobile devices equipped with acceleration sensors are very suitable for 3D rotation tasks.

Keywords: Tabletop, Multitouch, 3D User Interface, Mobile 3D Interaction, Smartphone, User Centered Design.

1 Introduction

Since the first Microsoft Surface appeared on the market in 2008, tabletop devices got quite famous and a lot of effort has been spent in technical improvements and the development of new interaction paradigms. It has been shown that these devices are suitable for special use cases like rapid prototyping [1], architecture and city / landscape planning [2,3], disaster management [4], collaborative document management [5] as well as training and learning [6,7]. In general terms it has been shown that tabletop devices in combination with Natural User Interfaces can dramatically enhance collaborative scenarios. One reason for this is the size of the display. It allows every participant in a collaborative setting to see WHO is currently doing WHAT. This so-called awareness of others [8,9] can't be found in classical Computer Supported Cooperative Work scenarios if multiple private devices are used instead of a large shared display. When developing collaborative applications for tabletop devices designers and programmers have to consider several aspects like the orientation of graphical objects [10], private and public spaces [11], security issues and group dynamics [12]. We have shown that User Centered Design (UCD) is a

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 100–114, 2013.

powerful tool to analyze requirements, workflows and needs of potential end users from the very beginning resulting in a high degree of usability and user satisfaction [4]. UCD experts already developed suitable methods like questionnaires, interviews and studies that are also suitable for the analysis of tabletop applications. Since technology already enables the tracking of multiple fingers as well as objects on a tabletop surface, various kind of interaction technologies have been developed. However, a single finger only allows the control of two degrees of freedom (DOF) of a graphical object on a flat surface (the x and y position) so that its interaction capabilities are limited. Multitouch gestures can help to overcome this limitation. However, using more than two fingers (controlling more than four degrees of freedom) results in lesser efficiency and a higher error rate [13]. Tangible objects equipped with optical markers that are lying on top of a tabletop device offer three degrees of freedom (x and y position as well as z-rotation). The drawback here is that objects have to be placed on the tabletop all the time for continuous tracking, resulting in a high degree of occlusion.

Our idea is to use sensor equipped mobile devices to control graphical objects on a tabletop device. Since such kind of active tangible object should be easily accessible by everyone we decided to use smartphones. Smartphones are wide spread personal devices and they consist of a variety of embedded sensors like gps, accelerometers, gyroscopes, proximity sensors and touchscreens. These sensors can easily be used for the analysis of different kind of 2D or 3D gestures. Further on, smartphones offer network access and therefore the possibility for data exchange with peripheral devices like other smartphones or tabletop computers. Another big advantage is that smartphones are personal storages and the display can perfectly be used as a personal area in collaborative scenarios, which also fulfills data security requirements. The goal of the ongoing research presented in this paper is therefore to create concepts for 3D interaction techniques with mobile devices within a collaborative tabletop scenario. We decided to perform an iterative UCD process using questionnaires, user performance tests and behavior observations.

The paper is structured as follows: First we present relevant related work in the area of 3D interactions and visualizations on tabletops as well as the combination of smartphones and tabletop devices. Next, we present our ideas of how to interact with 3D data on a tabletop using smartphones. In section four we compare these techniques with multitouch and tangible interaction techniques in two early user studies. Finally we sum up and describe ideas for future work.

2 Related Work

This section introduces relevant related work from the areas of 3D tabletop interaction and the combination of tabletops and mobile devices like smartphones.

2.1 3D Interaction and Visualization on Tabletops

Having a look at multitouch related literature the direct coupling is often mentioned as one of the biggest advantages. Using 2D graphical user interfaces, one or two fingers can be used to drag, rotate and scale a virtual object while every finger remains on the same position on it. However, using 3D user interfaces, a touchpoint may drift over

the object due to the perspective projection and 2D input combined with 3D output [14]. Hancock et al. introduced Sticky Tools [15]. A 3D gesture set that allows the usage of one up to three fingers to control graphical 3D objects while the virtual objects do not drift but remain stuck on the user's fingers. The classical pinch and spread gestures are in this case used to perform depth translations. However, as already mentioned, two fingers are not sufficient to perform six DOF interactions for full 3D translation and rotation. Therefore three finger interactions have also been realized within Sticky Tools. However, these gestures are only suitable for large screens and hardly useable on smartphones.

Similar approaches have been developed by Martinet et al. [16], but here only translations are considered. Reisman et al. [17] present a solution where also three fingers are used to control six DOF. However, according to Han et al. [5] we already showed that it is often easier and more efficient to use widgets or buttons for 3D control [18] even though the naturalness and the direct coupling of touch are lost. Visualizing 3D data on multiple screens and windows allows multiple perspectives to be rendered. Martinet et al. showed that the sequent control of just a few DOF on multiple different viewing perspectives could result in a simple and direct control of six DOF [16,19]. For the visualization of depth transparency and shadows have been proven to be suitable solutions [20]. It has to be mentioned that without using stereoscopic display technology it is only possible to simulate objects that are inside of a tabletop device.

With tangible objects on interactive tabletops users can control three degrees of freedom at once (x, y position and z rotation, see figure 1 center and right). Hence, using two objects or a state switch, all six DOF are controllable. With *dSensingNI* we already presented an approach of tracking objects in 3D space above a tabletop device [21]. This approach has the advantage that arbitrary physical objects can be used. However, as already mentioned, in collaborative settings it is reasonable to offer private screens and storage to the user. Smartphones could be tracked using depth-sensing cameras and *dSensingNI* but due to body occlusions when interacting with the device near to the human body, integrated sensor technology is much more reliable and precise.

2.2 Smartphone Interactions with Tabletops

Until now smartphones in combination with tabletop computers have mainly been used to transmit data from the private storage to the public available tabletop interface. Cuypers et al. present an approach where the embedded camera of a smartphone that is placed on the tabletop surface is used to detect its position and therefore it can be used as a classical tangible object [22]. The UbiTable [11] introduced the combination of mobile devices and tabletops using public working areas on the table, private areas on the mobile device and private areas on the table likewise. This setting allows for multiple kinds of collaborative workflows while considering data security and multiuser interaction conflicts.

The Bluetable [23] can establish a Bluetooth connection between a smartphone and a tabletop. Therefore the table detects the shape of the phone via computer vision algorithms before user authentication takes place. The authors further on present a simple photo-sharing application that makes use of this technology.

A similar scenario was realized in the PhoneTouch project [24]. Here trigger gestures are realized. If a user places a corner of the smartphone on the tabletop surface this is recognized via touchpoint detection and also by the phone itself by analyzing the accelerometer data. Therefore detection and identification can be performed automatically.

2.3 Summary

Until now much effort has been spend in developing interaction technologies for tabletop devices that allow the control of six DOF in order to interact in 3D space. Multitouch as well as tangible interaction using arbitrary physical objects have been studied. Smartphones have been used to create private workspaces, to transmit private data to a collaborative setting and to identify devices and users. However, to the knowledge of the authors there is no project existing that uses smartphones as private storage, private workspace and interaction device in combination with 3D user interfaces on a tabletop. We think that there is a huge potential for such scenarios. Therefore we developed multimodal interaction techniques that are introduced in the next section and performed two early user studies to evaluate our ideas within an architecture use case scenario.

3 Multimodal and Mobile 3D Tabletop Interaction

As already stated, there are important use cases that require a visualization of 3D content within a collaborative tabletop scenario. Yu et al. report the high degree of immersion and the possibility to use natural skills for interaction [14]. In order to find the best possibilities to control 3D data and to support group dynamic and workflows we created three different kinds of interaction techniques. We first developed techniques for multitouch as well as for tangible objects that are placed on the tabletop (this technique is called *tangible interaction* in the following sections) in order to control 3D objects. After that we built a prototype that uses smartphones. Since a smartphone additionally offers a private screen to its user, we think that this type of interaction has a very huge potential. However, it has to be investigated whether the control of six DOF is realizable with a smartphone device and if interaction metaphors can be found that are intuitively useable.

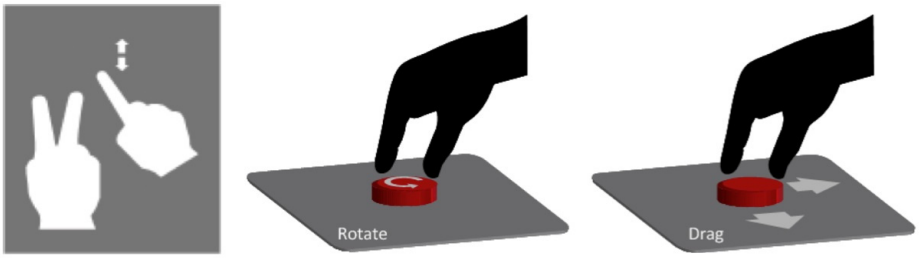


Fig. 1. Multitouch and Tangible Interaction Controlling Multiple Degrees of Freedom

3.1 Multitouch and Tangible Interaction

Our 3D multitouch concept is based on the Opposable Thumb approach presented by Hancock et al. [15]. Using one single finger a user can translate an object in the x/y plane[1]. Since we plan to implement scaling possibilities of 3D objects in the future we decided not to use the two-finger spread and pinch gestures for z-translation. Instead we implemented a three-finger approach. While the user touches a virtual object on the tabletop screen with two fingers she can move a third finger up and down for z-translation (see figure 1, left). This only works if the third finger is not touching the object. Using three fingers ON the object causes 3D rotations as described in the Opposable Thumb approach [15]. Z-rotation is performed using two fingers as known from many 2D multitouch applications. Hence, this approach allows the control of six DOF using one up to three fingers.

Our tangible interaction concept uses two physical objects to control a virtual 3D object. The physical objects are black cylinders with a height of 2 cm and a radius of 3.5 cm. Each cylinder is able to control three DOF: Two via x/y translation (drag) and one via z rotation (see figure 1, center and right). We decided to use one cylinder for translation and one for rotation only. The translation object controls the x/y position of the virtual 3D object via dragging and the z-position via rotation. The second cylinder controls the pitch angle via horizontal dragging, the roll angle via vertical dragging and the yaw angle via rotation.

Since tangible interaction results in a high degree of occlusion of the display we decoupled the virtual and real objects. Therefore a cylinder can be placed at an arbitrary position on the tabletop surface. Hence, every cylinder is bound to a specific virtual 3D object and to a specific functionality (either translation or rotation).

For the tracking of fingers and objects we use FTIR and DI infrared illumination on our tabletop device *useTable*[2]. The image analysis is done by reacTIVision[3] and CCV[4].

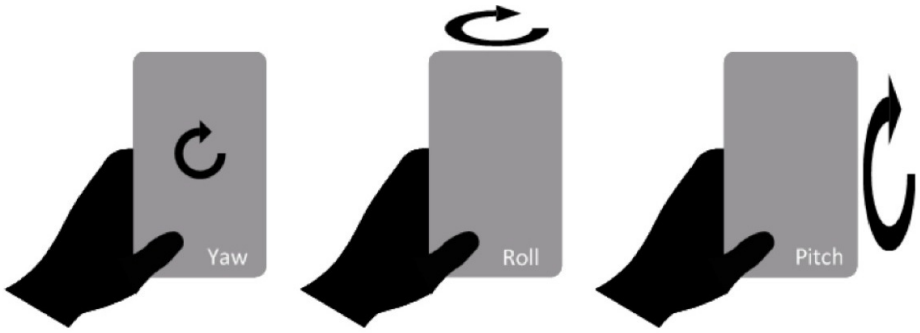


Fig. 2. 3D Orientation Angles of the Smartphone Measured by Accelerometers

[1] In the context of this paper the x/y plane is defined by the tabletop surface while the z-axis extends into the table.

[2] `http://www.usetable.de`

[3] `http://reactivision.sourceforge.net/`

[4] `http://ccv.nuigroup.com/`

3.2 Using Smartphones as 3D Input Devices

The embedded sensors of a smartphone offer different possibilities to control the six DOF of a virtual 3D object. Within the ongoing research project presented in this paper we plan to evaluate different approaches within a UCD process. Our initial idea is to mainly use accelerometers that offer three values for 3D orientation (see figure 2). Since the touchscreen of the smartphone is going to be used as personal display it is obvious that is can also be used for data input. For example a button could be realized that switches between translation and rotation control. Hence, the three accelerometers and the button could be used for six DOF control. The applied interaction concept will be explained in the upcoming evaluation section.

For the prototype realization we used Windows Phone smartphones[5] and created an extendable software framework that handles the communication between the mobile device and the tabletop. It is called SmartPhoneControl-Framework. For the transmission of sensor data that is used to control 3D objects we used the TUIO [25] protocol. We created a new profile /tuio/STobj that offers data fields for acceleration as well as 3D rotation using quaternions. Further on predefined states and gestures that are interpreted on the device side can be transmitted using two newly created fields.

In order to enable a bidirectional communication of rich data we also implemented the XMPP protocol. It may be used to send feedback to the mobile device, personal documents or 3D objects to the table. Figure 3 shows a scheme of the frameworks hard- and software components.

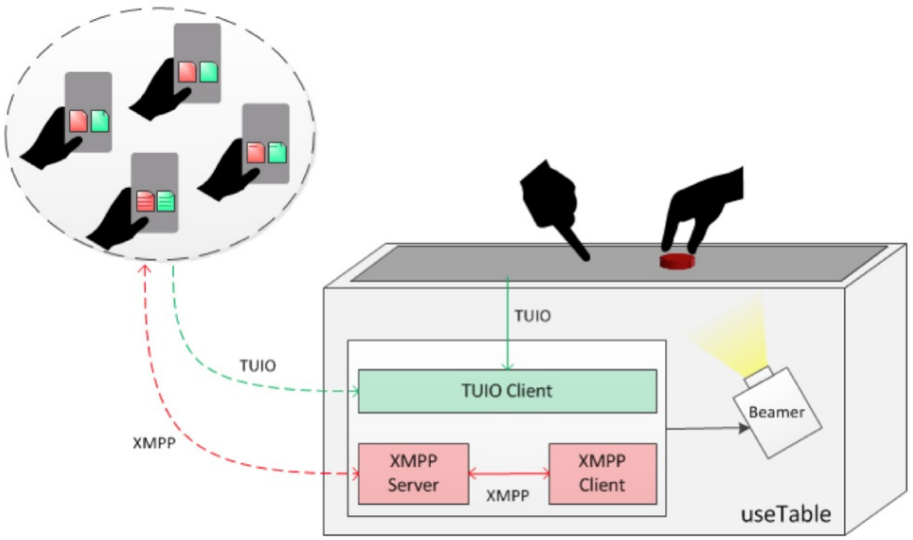


Fig. 3. SmartPhoneControl Framework

[5] We use a Samsung Omnia 7, a LG Optimus 7 and a HTC Mozart.

4 Evaluation

In this section we present two early user studies we held in our laboratory in order to compare the previously described interaction techniques and to evaluate the group work. Even though the number of participants is low the results show interesting trends and we gathered important think-alouds from the users.

4.1 Single User Study

The goal of the first user study was to analyze the effort and precision of the interaction techniques as well as the user experience. Therefore we asked five colleagues from our laboratory to perform a docking test and answer an INTUI [26] questionnaire. This questionnaire allows an evaluation of the intuitiveness of different interaction techniques. All questions have to be answered on a scale from 1 (very low) to 7 (very high).

Fig. 4. Docking Task in the Single user Study

In the docking test the users had to dock a virtual armchair onto another one (see figure 4). The test consisted of six tasks. It began very simple but got more complicated after successfully completing a task. In Task 1 (T1) users only had to translate the object along the x axis (1 DOF), in T2 users had to translate the object along the x and the y axis (2 DOF), in T3 users had to move the object along the z axis (1 DOF), in T4 users had to rotate the object around the z axis (1 DOF), in T5 users had to rotate around all three axes (3 DOF) and finally in T6 users had to control all six DOF. Every participant had to use all three interaction techniques but in an alternating order. The smartphone device offered the possibility to translate and rotate the object via the accelerometer sensor values and a switch button (translation/rotation) on the touchscreen. All task completion times can be found in

tables 1 to 3. Some users had not been able to complete a task. This is indicated with a ”-” in the tables.

It can be seen that for the simple tasks (T1, T3) multitouch and tangible interaction are very fast. This might be caused by the fact that the smartphone interaction requires learning while the other tasks are more or less known from state of the art approaches. In T2 it took the users very long to complete the z-translation using multitouch. This can also be interpreted as a learning artifact because we invented a new three-finger technique as described previously. In the rotation tasks (T4, T5) the smartphone interaction was fastest. This shows that using accelerometer sensor data is very effective for rotation tasks. In the combined task T6 multitouch and tangible interaction are again faster, mainly caused by the very good translation performance. However the standard deviation values of T6 show that completion times vary strongly. Analyzing the INTUI questionnaire answers (tables 4 to 6) it can be seen that users rated the learning effort of the multitouch interaction highest (4.0, tangible interaction: 2.0, smartphone: 2.2). We think that this is mainly caused by the complicated three finger interaction techniques. Users further on rated the precision of the smartphone highest (4.8, multitouch: 2.6, tangible interaction: 4.6). Also the fun factor (enjoyment) was rated best with the smartphone approach (5.6, multitouch: 5.0, tangible interaction: 5.4). The INTUI results (visualized in figure 6) have large deviation values therefore this results can only be seen as first trends but further studies including a larger number of participants have to be performed. However, these early results helped us designing a multiuser application for the collaborative design of an office lobby. Here our hypothesis was that smartphones are very suitable due the embedded storage and the private display. The interaction techniques for the smartphones have been slightly adapted according to the results of the single user study. The application as well as the results of a multiuser study are presented in the next section.

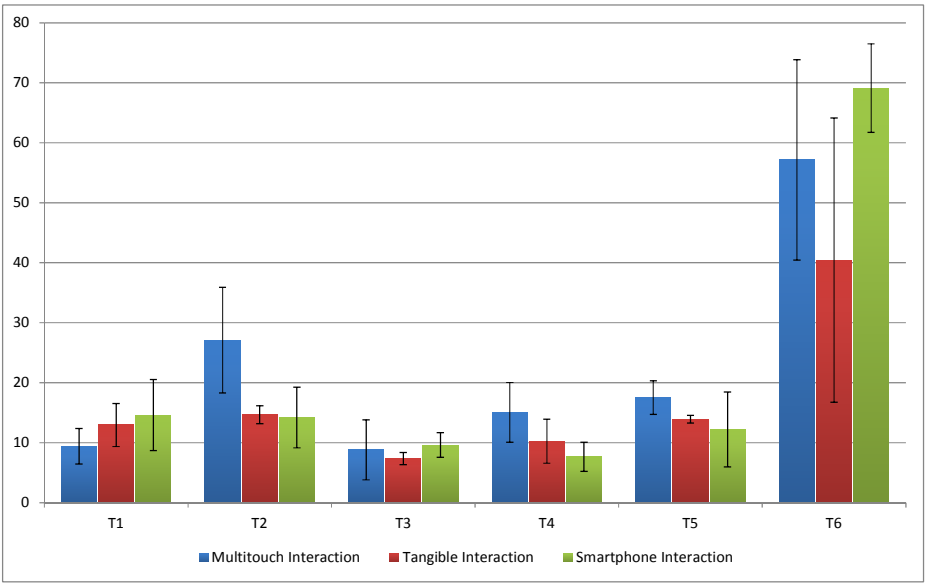


Fig. 5. Average Task Completion Times and Standard Deviations

Table 1. Task Completion Times for Multitouch Interaction (Values in Seconds)

Participant	T1	T2	T3	T4	T5	T6
1	6.9	17	17.3	13.7	17.9	80
2	5.9	37.6	5.4	14.6	14.9	-
3	11.5	20.3	8.8	-	-	44.9
4	12.8	34.2	7.7	21.9	21.3	44.5
5	9.9	26.4	4.9	10	15.9	59.2
Avg. Time	9.40	27.09	8.80	15.04	17.51	57.15
Std. Deviation	2.95	8.80	5.00	4.99	2.81	16.70

Table 2. Task Completion Times for Tangible Interaction (Values in Seconds)

Participant	T1	T2	T3	T4	T5	T6
1	9.7	13.4	6.9	13.7	14.4	32.0
2	8.9	16.5	6.6	14.2	-	-
3	13.9	16.0	7.5	8.8	13.2	75.3
4	14.6	13.3	6.6	5.6	14.1	32.3
5	17.5	14.0	9.0	8.9	-	22.2
Avg. Time	12.92	14.64	7.31	10.23	13.90	40.45
Std. Deviation	3.58	1.50	1.02	3.66	0.62	23.70

Table 3. Task Completion Times for Smartphone Interaction (Values in Seconds)

Participant	T1	T2	T3	T4	T5	T6
1	9.1	7.9	11.0	7.0	15.2	-
2	13.1	13.6	12.1	11.6	17.9	73.6
3	23.0	20.9	7.9	6.2	12.2	60.0
4	13.1	17.2	9.9	8.0	-	76.4
5	-	11.3	7.2	5.4	3.5	66.5
Avg. Time	14.58	14.17	9.62	7.64	12.20	69.13
Std. Deviation	5.92	5.06	2.05	2.41	6.25	7.37

Table 4. Questionnaire Results for Multitouch Interaction

Participant	Motion Effort	Learn Effort	Precision	Enjoyment
1	2	4	1	4
2	5	2	3	5
3	6	6	2	3
4	5	6	2	7
5	2	2	5	6
Avg.	4.0	4.0	2.6	5.0
Std. Deviation	1.87	2.00	1.52	1.58

Table 5. Questionnaire Results for Tangible Interaction

Participant	Motion Effort	Learn Effort	Precision	Enjoyment
1	2	1	7	6
2	4	3	3	5
3	2	2	6	5
4	2	2	1	5
5	3	2	6	6
Avg.	2.6	2.0	4.6	5.4
Std. Deviation	0.89	0.71	2.51	0.55

Table 6. Questionnaire Results for Smartphone Interaction

Participant	Motion Effort	Learn Effort	Precision	Enjoyment
1	2	2	4	6
2	2	2	3	3
3	1	1	6	6
4	6	4	5	6
5	1	2	6	7
Avg.	2.4	2.2	4.8	5.6
Std. Deviation	2.07	1.10	1.30	1.52

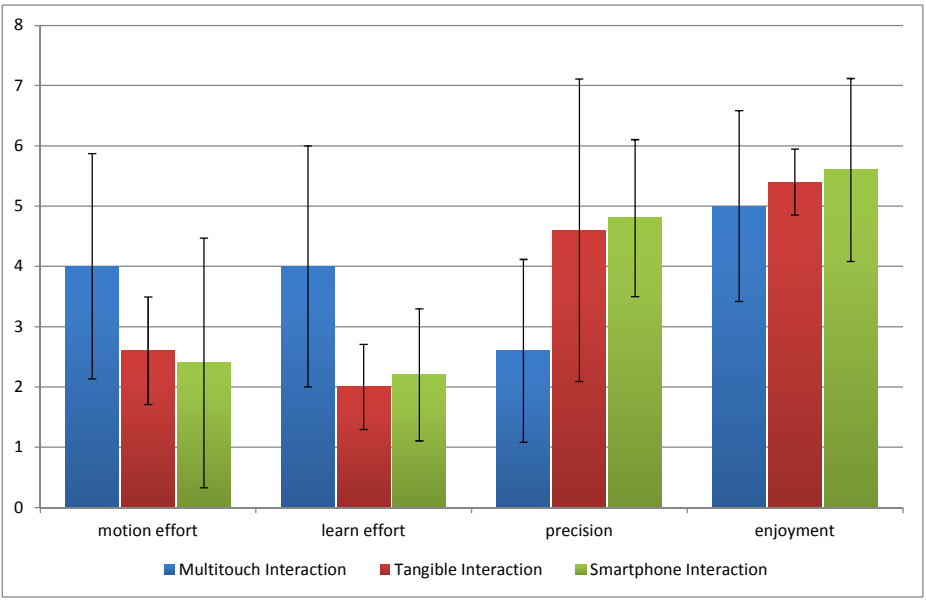


Fig. 6. Results of the INTUI Questionnaire

4.2 Multiuser Study

After getting first insights about the quality of the developed interaction techniques we applied some improvements based on the measurements, questionnaires and recorded think-alouds. For example we realized the possibility to change the camera perspective of the 3D scene. This enables a translation with one finger and rotation with two fingers to control all six DOF since the affected axes can be switched. So no three-finger techniques are required anymore (even though still working). Further on the smartphone interface now enables the rotation around a single axis by disabling the other two. The task in the second study was to collaboratively design a lounge for an office. 15 people participated in this study divided into five groups, three participants each. All users got a short training of all three interaction techniques and five minutes time to get familiar with them. Then the design task was assigned to the participants and we asked them again to think-aloud. We used video and audio recording for later analysis and after completing the task every user had again to fill out an INTUI questionnaire. The participants got a smartphone as personal display and personal interaction device. Via the smartphone and our framework the users were able to add new pieces of furniture to the scene and to remove them. Figure 7 shows the 3D scene and figure 8 three users with their smartphones designing a lounge. Figure 9 shows the personal display of the smartphone (left: select the kind of interaction performed through the accelerometers, center: furniture menu, right: different perspectives of a selected armchair).

The most interesting result of the multiuser study is that nearly all users mainly decided to use the smartphone for 3D interaction. We believe that this is caused by two facts: First, the users had to use the smartphone to chose a furniture they like to add to the scene via the smartphone display, therefore it is obvious to seamlessly continue the use of the device for object manipulation. Second, the user experience with this kind of interaction is really good, proven by the fact that after the experiencing phase the users rated the smartphone interaction as the easiest one.

Fig. 7. Virtual 3D Scene of an Office Lounge

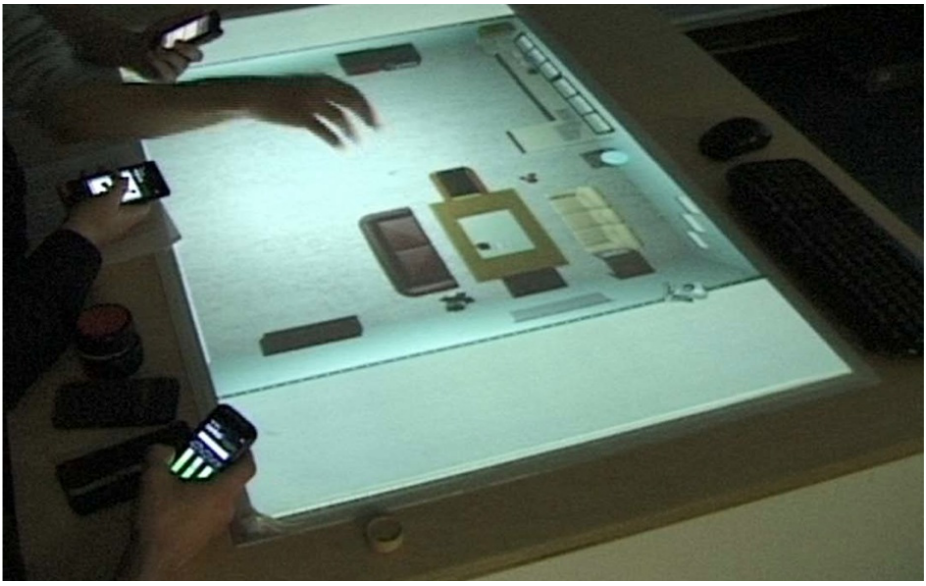


Fig. 8. Collaborative Design on the useTable using Smartphones, Multitouch and Tangible Interaction

Fig. 9. Private Display of the Smartphone

We observed that there was much discussion going on during the design task. Often users discussed general ideas before they split the virtual room into three regions. After that they proceeded individually and each user designed one region on her own. This single user process was often shortly interrupted by group discussions about the general ideas including critical comments about already completed things. Another interesting aspect is that even though it was introduced at the beginning only

one group decided to make use of the possibility to switch camera perspectives. We observed that many users tried to change their position on the table to get a different perspective but then recognized that this had no effect on the 3D perception.

All users who tried the manipulation with tangible objects switched back to smartphone interaction after a short while. They all moaned about the occlusion caused by the objects. In their think-alouds the participants presented good ideas for future improvements. E.g. one user asked for a reset button after his piece of furniture got lost in space. Another user recommended the use of a physic engine for collision detection to make the 3D interaction easier and more natural. A third user asked for the possibility to rotate in steps of 90 degrees since this is hard to do precisely with the implemented techniques. Further on he criticized that it is not possible to have more than one instance of a piece of furniture in the 3D scene.

The result of the INTUI questionnaire showed an overall intuitiveness value of 5.27. The fun factor of the application was rated with an average value of 5.8. This shows that with our smartphone approach we are on the right track to intuitive 3D interaction on tabletop devices that shows high usability.

5 Conclusion

In this paper we presented an ongoing research on 3D interaction techniques for tabletop devices. In two early user studies we evaluated multitouch, tangible interaction and especially sensor-equipped smartphones as external devices. The results show that multitouch 3D interaction requires a huge cognitive effort while tangible objects cause too much occlusion. Therefore we think that smartphones are very suitable and our study results show that they allow for easy, fast and intuitive usage, especially in rotation tasks. Even though virtual and real objects are no longer coupled in this approach, most participants rated the user experience of this indirect interaction as a very intuitive one. Further on the fun factor seems to be high.

Nevertheless, we got important insights for improvements. To support group work we also have to study the implications on the group awareness and dynamics our techniques cause. Therefore, after redesigning the interaction techniques and the overall workflows further studies and observations are needed in order to best fulfill all requirements for collaborative 3D tabletop scenarios. We also plan to integrate additional sensors of mobile devices like a digital compass.

References

1. Akaoka, E., Ginn, T., Vertegaal, R.: DisplayObjects: prototyping functional physical interfaces on 3d styrofoam, paper or cardboard models. In: Fourth International Conference on Tangible, Embedded, and Embodied Interaction (TEI), pp. 49–56. ACM Press, New York (2010)
2. Ishii, H., Ratti, C., Piper, B., Wang, Y., Biderman, A., Ben-Joseph, E.: Bringing Clay and Sand into Digital Design - Continuous Tangible user Interfaces. BT Technology Journal 22, 287–299 (2004)

3. Piper, B., Ratti, C., Ishii, H.: Illuminating clay: Illuminating Clay: A 3-D Tangible Interface for Landscape Analysis. In: SIGCHI Conference on Human Factors in Computing Systems, pp. 355–362. ACM Press, New York (2002)
4. Nebe, K., Klompmaker, F., Jung, H., Fischer, H.: Exploiting New Interaction Techniques for Disaster Control Management Using Multitouch-, Tangible- and Pen-Based-Interaction. In: Jacko, J.A. (ed.) Human-Computer Interaction, Part II, HCII 2011. LNCS, vol. 6762, pp. 100–109. Springer, Heidelberg (2011)
5. Nebe, K., Müller, T., Klompmaker, F.: An Investigation on Requirements for Co-located Group-Work Using Multitouch-, Pen-Based- and Tangible-Interaction. In: Jacko, J.A. (ed.) Human-Computer Interaction, Part II, HCII 2011. LNCS, vol. 6762, pp. 90–99. Springer, Heidelberg (2011)
6. Kaschny, M., Buron, S., von Zadow, U., Sostmann, K.: Medical education on an interactive surface. In: ACM International Conference on Interactive Tabletops and Surfaces (IST), pp. 267–268. ACM Press, New York (2010)
7. Conradi, B., Hommer, M., Kowalski, R.: From digital to physical: learning physical computing on interactive surfaces. In: ACM International Conference on Interactive Tabletops and Surfaces (IST), pp. 249–250. ACM Press, New York (2010)
8. McCrindle, C., Hornecker, E., Lingnau, A., Rick, J.: The design of t-vote: a tangible tabletop application supporting children's decision making. In: 10th International Conference on Interaction Design and Children, pp. 181–184. ACM Press, New York (2011)
9. Hornecker, E.: Tangible User Interfaces als kooperationsunterstützendes Medium, Doctoral Thesis, Elektronische Bibliothek, Staats und UniversitŁtsbibliothek Bremen (2004)
10. Shen, C., Ryall, K., Forlines, C., Esenther, A., Vernier, F.D., Everitt, K., Wu, M., Wigdor, D., Morris, M.R., Hancock, M., Tse, E.: Informing the Design of Direct-Touch Tabletops. IEEE Comput. Graph. Appl. 26, 36–46 (2006)
11. Shen, C., Everitt, K., Ryall, K.: UbiTable: Impromptu Face-to-Face Collaboration on Horizontal Interactive Surfaces. In: Dey, A.K., Schmidt, A., McCarthy, J.F. (eds.) UbiComp 2003. LNCS, vol. 2864, pp. 281–288. Springer, Heidelberg (2003)
12. Morris, M.R., Cassanego, A., Paepcke, A., Winograd, T., Piper, A.M., Huang, A.: Mediating Group Dynamics through Tabletop Interface Design. IEEE Comput. Graph. Appl. 26(5), 65–73 (2006)
13. Kin, K., Agrawala, M., DeRose, T.: Determining the benefits of direct-touch, bimanual, and multifinger input on a multitouch workstation. In: Graphics Interface, pp. 119–124. Canadian Information Processing Society (2009)
14. Yu, L., Svetachov, P., Isenberg, P., Everts, M.H., Isenberg, T.: FI3D: direct-touch interaction for the exploration of 3D scientific visualization spaces. IEEE Trans. Vis. Comput. Graph. 16(6), 1613–1622 (2010)
15. Hancock, M., ten Cate, T., Carpendale, S.: Sticky tools: full 6DOF force-based interaction for multi-touch tables. In: ACM International Conference on Interactive Tabletops and Surfaces (IST), pp. 133–140. ACM Press, New York (2009)
16. Martinet, A., Casiez, G., Grisoni, L.: The design and evaluation of 3D positioning techniques for multi-touch displays. In: IEEE Symposium on 3D User Interfaces, pp. 115–118. IEEE Press, New York (2010)
17. Reisman, J.L., Davidson, P.L., Han, J.Y.: A screen-space formulation for 2D and 3D direct manipulation. In: 22nd Annual ACM Symposium on User Interface Software and Technology (UIST), pp. 69–78. ACM Press, New York (2009)

18. Klompmaker, F., Nebe, K.: Towards 3D Multitouch Interaction & Widgets. In: Workshop on Interaction Techniques in Real and Simulated Assistive Smart Environment, Workshop at the First International Joint Conference on Ambient Intelligence, AmI (2009)
19. Martinet, A., Casiez, G., Grisoni, L.: 3D positioning techniques for multi-touch displays. In: 16th ACM Symposium on Virtual Reality Software and Technology (VRST), pp. 227–228. ACM Press, New York (2009)
20. Hilliges, O., Izadi, S., Wilson, A., Hodges, S., Garcia-Mendoza, A., Butz, A.: Interactions in the air: adding further depth to interactive tabletops. In: 22nd Annual ACM Symposium on User interface Software and Technology (UIST), pp. 139–148. ACM Press, New York (2009)
21. Klompmaker, F., Nebe, K., Fast, A.: dSensingNI - A Framework for Advanced Tangible Interaction using a Depth Camera. In: Sixth International Conference on Tangible, Embedded, and Embodied Interaction (TEI), pp. 217–224. ACM Press, New York (2012)
22. Cuypers, T., Francken, Y., Vanaken, C., van Reeth, F., Bekart, P.: Smartphone Localization on Interactive Surfaces Using the Built-in Camera. In: Procams, p. 448 (2009)
23. Wilson, A.D., Sarin, R.: BlueTable: connecting wireless mobile devices on interactive surfaces using vision-based handshaking. In: Graphics Interface, pp. 119–125. ACM Press, New York (2007)
24. Schmidt, D., Chehimi, F., Rukzio, E., Gellersen, H.: Phone- Touch: a technique for direct phone interaction on surfaces. In: 23nd Annual ACM Symposium on User Interface Software and Technology (UIST), pp. 13–16. ACM Press, New York (2010)
25. Kaltenbrunner, M., Bovermann, T., Bencina, R., Costanza, E.: TUIO - A Protocol for Table Based Tangible User Interfaces. In: 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (2005)
26. Ullrich, D., Diefenbach, S.: INTUI. Exploring the Facets of Intuitive Interaction. In: Mensch & Computer, p. 251. Oldenbourg Verlag (2010)

Multimodal Mobile Collaboration Prototype Used in a Find, Fix, and Tag Scenario

Gregory M. Burnett[1], Thomas Wischgoll[2], Victor Finomore[1], and Andres Calvo[3]

[1] Air Force Research Laboratory, 711 Human Performance Wing, WPAFB, OH
[2] Deptment of Computer Science and Engineering Wright State University, OH
[3] Ball Aersopace, Fairborn, OH
{gregory.burnett,victor.finomore,andres.calvo.ctr}@wpafb.af.mil,
thomas.wishgoll@wright.edu

Abstract. Given recent technological advancements in mobile devices, military research initiatives are investigating these devices as a means to support multimodal cooperative interactions. Military components are executing dynamic combat and humanitarian missions while dismounted and on the move. Paramount to their success is timely and effective information sharing and mission planning to enact more effective actions. In this paper, we describe a prototype multimodal collaborative Android application. The mobile application was designed to support real-time battlefield perspective, acquisition, and dissemination of information among distributed operators. The prototype application was demonstrated in a scenario where teammates utilize different features of the software to collaboratively identify and deploy a virtual tracker-type device on hostile entities. Results showed significant improvements in completion times when users visually shared their perspectives versus relying on verbal descriptors. Additionally, the use of shared video significantly reduced the required utterances to complete the task.

Keywords: Multimodal interfaces, mobile computing, remote collaboration.

1 Introduction

Battlefield Airmen (BA) serve a myriad of roles and responsibilities in an ever-changing austere battlespace. BA are equipped with wearable communication technologies to assist them in their mission objectives, enhance situation awareness, and provide interoperable means to communicate with distributed entities. Moreover, these mobile technologies are utilized to capture and disseminate battlefield information to distributed teammates to support dynamic decision-making (DDM).

Although BA are highly trained and possess a variety of skills, they can confront situations that are outside their expertise. Often the operator is left to complete the task on their own or must wait for an expert to arrive on scene, both unfavorable solutions in time-critical situations. The multimodal software approach described in this paper seeks to foster the ability of the dismounted operator to remotely collaborate with a subject matter expert (SME) on physical tasks. Kraut et al. [1]

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 115–128, 2013.

defines remote collaboration on a physical task as *"a general class of 'mentoring'… in which one person directly manipulates objects with the guidance of one or more other people, who frequently have greater expertise about the task". (p.16).* Task knowledge is important to enable remote collaboration; however to effectively collaborate, Clark et al [2] report there needs to be a mutual understanding between helper and worker to ensure *common ground*. This common ground can be achieved through the use of shared perspectives afforded by mobile device technology.

Advancements in mobile device hardware and software support DDM and foster remote collaboration. These improvements offer great potential to improve dismounted personnel's mission effectiveness, particularly during dynamic re-planning and mission execution. Mission performance and outcomes are often inversely correlated to time constraints resulting from real-time decision making [3]. Some examples include: survivability of time-critical causalities being attended to by infield medics through the guidance of a remote surgeon; battlefield repair of machinery by the end users being advised by a non-collocated mechanic; defusing improvised explosive device (IED) ordinance by untrained soldiers under the expert guidance of highly trained ordinance disposal personnel.

As military conditions and environments change so must the equipment used by BA. Within the last several years, there has been a research and technology development shift in wearable computing for combat ground personnel. Previous investments were spent miniaturizing rugged laptops and funding ultra-mobile personal computers development. Today, research focuses on leveraging and adapting emerging smartphone and handheld tablet devices for battlefield applications.

In addition to technical advancements, smartphones and handheld tablets are increasingly becoming more user friendly, with easy to use interface controls and advance features. Out-of-the-box, these devices' I/O mechanisms are designed to be interacted with while on the move. In addition, the packaging and flexible form factor of these devices enables them to be easily integrated into BA's personal battlefield ensemble. Another significant advantage is their lighter weight compared to previously fielded computing platforms. Lastly, mobile operating systems are aggressively improving power management with customizable power consumption features and settings. Leveraging the new mobile devices' vast and ubiquitous capabilities, research initiatives within the Air Force Research Laboratory (AFRL) are beginning to investigate their applicability to support multimodal mobile remote collaboration for BA and design intuitive interfaces to enhance mission effectiveness**.**

Smartphones and handheld tablets adaptation into warfighter ensembles depends on their intrinsic capabilities to function in multimodal communicative roles. It is often the case that dismounted BA must divide their cognitive resources across several tasks simultaneously. Audio, visual, and/or tactile perception channels may be masked or already occupied, rendering certain modalities ineffective for receiving additional information. Accordingly, special attention should be given in designing a mobile application that is flexible in its communication capabilities and modalities making it conducive to the extreme tempo and interaction of the dismounted military forces.

In this paper, we describe the implementation and design of an Android mobile application that facilitates multimodal remote collaboration. We first discuss previous related work and highlight features and capabilities that warrant inclusion into our application's implementation. These features seek to enhance greater communication grounding between teammates working cooperatively to complete a common task. Next, we report the methods and findings from a demonstration of the mobile application, where participants executed a joint find, fix, and tag scenario. Finally, future work and discussion sections are provided. These sections document enhancements to the developmental mobile application and discuss military benefits from real-time decision making capabilities supported by mobile collaboration.

2 Related Work

There are numerous technological approaches, multimodal apparatuses, and devices used to facilitate distributed collaboration. In the following section, we review several of these devices and highlight their design features and results.

Several researchers sought to use shared video perspectives to aid in collaboration of remote tasks. Kuzuoka et al [4] developed a spatial workspace collaboration system that enabled remote helpers to visually monitor the active workspace of the worker via a live video feed captured by a head-mounted camera worn by the worker. Moreover, the worker was able to visually receive guidance through a head mounted display that rendered the helper's physical actions recorded from a camera focused on the helper. Kraut et al. [1] leveraged a worker worn capturing setup consisting of a camera and a heads-up display, as well as audio headsets to provide verbal guidance to a worker. These remote collaborating apparatuses using shared visual information have been shown to decrease task completion times with greater task accuracy. Fussell et al [5] suggests that "visual information facilitates grounding or the development of mutual understanding between conversational participants."

Real-time markup annotation of shared images and live video between helper and worker has been shown to improve performance. Ou et al's [6] remote collaboration DOVE (Drawing Over Video Environment) system allows a remote helper to annotate a video feed with free-form and gesture-fitting markups while providing task instructions. Their findings reported markup capability "significantly reduces performance time compared to camera alone." Stevenson et al's [7] research utilized a combination of "on-video" and "in-workspace" annotation capability, where remote helpers could use illustrated guidance to direct the action of the worker. The use of annotation techniques reduced the spoken instructions into "spoken fragments like 'in', 'out', 'around', and 'here' as they drew" their remote directives.

Performance assessments of a collaborative system and/or application can be evaluated using several metrics such as instructional response time, task completion time, and task accuracy. Thus, the effectiveness of our Android application, designed to aid in the distributed decision making of cooperative teammates, will be assessed using these metrics.

The overarching goal of these research efforts and our Android application is to support remote collaboration through various mobile capabilities and modalities, establishing a shared mental awareness of the cooperative tasks in an effective and timely manner.

3 Design

The prototype Android application described herein is part of an on going program internal to AFRL's Human Effectiveness Directorate. The program develops advanced wearable information management and cognitive interface technology for BA. The mobile application's implementation was based on a user-centered design approach and implemented a multimodal, context-rich distributed system, which enhances interactive collaboration of distributed participants.

The primary components of the system are a mobile device and a personal computer. The user of the mobile device is called the worker, and the user of the computer is called the helper. Although the primary use case we envision consists of dismounted BA as workers and task relevant BA SMEs as helpers, the system can be used in other situations, such as among BA as shown in Figure 1.

Fig. 1. Worker captures image of an explosive device with mobile devices and transmits image to Helper to obtain remote assistance

Drawing features from related work and understanding the dynamic mission conditions under which dismounted military personnel are required to execute, we formulated the following set of capabilities for our prototype mobile application:

- Sharing live video of the worker's workspace.
- Sharing full-duplex audio between linked users.
- Supporting markup annotation on captured still images.
- Overlaying annotated images on the worker's video feed with an adjustable transparency.
- Allowing the worker to switch between annotated image/live adjacent windows or merged annotated image and live perspective.

For the purposes of the prototype, we implemented data communication between an Android Samsung Galaxy Tablet and a personal computer using TCP/IP.

3.1 Sharing Live Video

Sharing a visual perspective of the active workspace is one of the fastest means to establish a common ground between cooperating individuals. Mobile devices are generally equipped with on-board cameras; however, depending on the circumstances, an off-board camera may be better suited for a collaborative task. Accordingly, our application was designed to accept a video capture device signal from either an embedded camera and/or an external camera tethered or wirelessly transmitting through TCP/IP. The degree of flexibility in video source(s) enables our mobile application to be scalable in order to address the demands and in-field capabilities.

Once a video capture device connection has been made live-video feed is transmitted to the helper as a series of 800x600 jpeg images at an average rate of 30 frames per second (fps). The mobile device displays a preview of the live-video feed as feedback for the worker. Implementing power conserving features, the frequency of image transmission is adjustable from 30 fps to 5 fps.

3.2 Sharing Audio

In addition to video communication, we implemented full-duplex audio communication across TCP/IP. The worker can choose to continuously transmit audio "hot mic" to the helper, or transmit audio only while depressing an external push-to-talk (PTT) button connected to the mobile device. Both input means were included in the design to address hands-free operations and for power consumption considerations. The mobile application can receive stereo or mono inputs and support a wide range of frequencies and sampling rates to accommodate the various military tactical communication headsets that may be used in conjunction with our system.

3.3 Still Image Capture and Annotation

Once the helper receives a live-video feed from the worker, he/she has the ability to capture a still image of the feed and annotate it with instructional content, as in Figure 2.

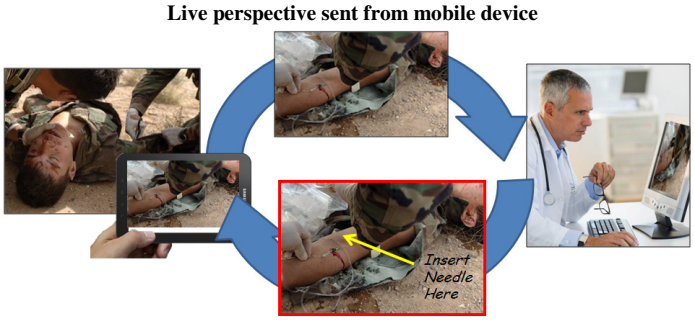


Fig. 2. Helper captures image and annotates it

Additionally, the worker may want to focus the attention of the helper to a particular area in the live-video and can likewise initiate markup annotation on the mobile device. The mobile application supports both free-form markups and predefined objects and symbols assisting markup execution. Assigning one of the mobile device hot keys, configurable in the source, to capture an image from the live-feed, the worker can then use touch screen interactions to create his/her desired additions to the image prior to transmitting it to the helper (see Figure 3).

Fig. 3. Worker annotates image

3.4 Adjustable Display Modes and Markup Transparency

Annotated images are presented to the worker in either of two display modes: full-screen or split-screen, as shown in Figure 4.

Fig. 4. Mobile application display modes

In split-screen mode, the worker sees the live-video feedback on the left half of the screen and the annotated image on the right half of the screen. In full-screen mode, the worker sees the live-video feed with a translucent overlay of the annotated image on the entire screen. The worker is able to toggle between split-screen and full-screen modes by pressing a button in the mobile device's options menu.

In full-screen mode, the worker adjusts the transparency of the overlay by using a pan gesture on the left half of the screen, as shown in Figure 5. Panning up reduces the transparency, and panning down increases the transparency. With these gestures, the worker can quickly set the transparency of the overlay to a level suitable for the current task. Note that the actions we chose to control the overlay's transparency level and toggle between display modes do not occupy any space on the user interface (UI) that would be used by the live-video feed or the annotated image. Thus, these controls do not get in the way of the live-video and annotated image.

Fig. 5. Markup transparency setting control

Workers can observe the annotated image in a way that they find suitable for the current task by switching between display modes and adjusting the overlay's transparency. For example, consider a scenario where the worker is a BA and needs to place an intravenous (IV) needle in a soldier's arm. The worker shows the helper, who has medical training, the live-video feed of the soldier's arm. The helper captures a still-image and annotates the best place to insert the IV needle. The worker can then set the display mode to split-screen to see where he/she needs to place the IV. Alternatively, the worker can set the display mode to full-screen with a medium transparency and aim the camera at the soldier's arm to obtain a precise guide for the task.

3.5 Data Exchange

Interfacing with tactical heterogeneous systems, the multimodal data captured by our application was not preprocessed with video/audio compression prior to transmission. Network and data optimization are handled by an intermediate network node between the cooperative helper/worker pair communicating with each other.

4 Evaluation

An initial demonstration involving a find, fix, and tag task was conducted to assess the extent to which the developed Android application supported remote collaboration. Participants communicated with a remote expert using various modalities to complete the evaluation task. Task components involved: 1) identify a specific individual from a crowd of people, 2) align aiming device on identified individual, and 3) initiate a tagging sequence. The modality interfaces investigated were Audio, Video with Markup, Video with Audio, and Video with Markup and Audio.

4.1 Participants

Eight military and four civilian participants volunteered for this study (eight men and four women) ranging in age from 23-30 (M =25) years. All participants had normal hearing and normal, or corrected-to-normal vision. Additionally, all participants had

prior training and experience handling a rifle. The participants collaborated with a remote confederate, who knew the sequence and identity of the individuals being tagged.

4.2 Experiment Design

A within-subject design that was balanced using a Latin-square procedure was employed with four levels of Modality Interface (Audio, Video with Markup, Video with Audio, and Video with Markup and Audio). All participants took part in a training session to familiarize themselves with the task and devices. The participants trained by communicating with the remote confederate and marking targets of interest with an AirSoft M-4 rifle per experimental condition. Participants were given the option for more practice trials; however, none of them felt the need for more. The four experimental conditions and virtual target configurations were randomized per participant.

4.3 Apparatus

Each participant (Worker) used an affixed pivoting AirSoft M-4 Rifle with a camera attached to the forward barrel as shown in Figure 6.

Fig. 6. Rifle with attached camera

Participants were instructed to stay behind a partition wall, which blocked their line of sight to the active scene, and utilize the rifle mounted camera's perspective for the task, as seen in Figure 7. The partition wall was positioned in front of an 8'x10' projection screen that rendered a virtual scene consisting of a gathering of 12 potential targets of interest.

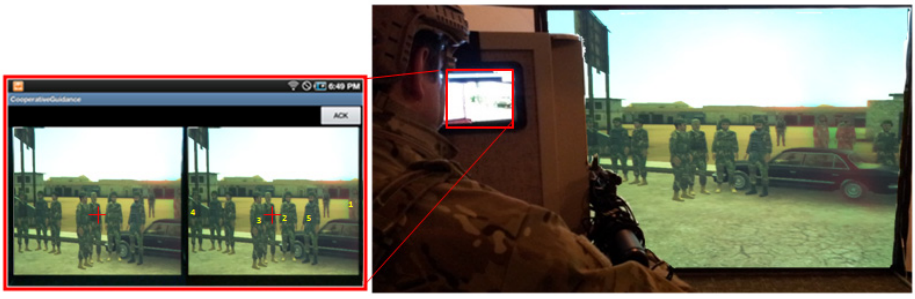


Fig. 7. Structure and experiment scene

The rifle/camera provided a live video feed to a Samsung Galaxy Tablet running our developmental Android application. The Tablet was stationary mounted to the partition wall allowing the participant to freely use their hands, as seen in Figure 7.

The remote SME (Helper) communicated with the participant through the tablet running the collaborative Android application through a Wi-Fi connection. They were situated in front of a workstation, which was isolated from the experimental area, as shown in Figure 8. The SME workstation allowed the cooperative pairs to communicate via streaming audio as well as capture and annotate still images from the participant's tablet. The SME used this tool to direct the participant in finding and tagging the hostiles in a specific order.

Fig. 8. SME collaborative workstation

4.4 Procedure

The four conditions that were evaluated included:

1) Audio only, where the SME could not see the participant's camera perspective and thus could to give directions via voice commands only.
2) Video with Markup, where the SME could monitor the participants' view from the camera mounted on the rifle and was able to annotate the still images, providing directives on the tablet.
3) Video with Audio, where the SME was able to monitor the participants' perspective and provide verbal directives.
4) Video with Markup and Audio, where the SME was able to monitor the participants' perspective and could provide directives through both markup and verbal interactions.

In the Audio condition, the SME had to verbally describe to the worker the characteristics of the individual that required tagging. The SME description of the

individual started with a clothing description, an indication of facial hair, and whether the individual was wearing anything on their head. The Video with Markup condition consisted of the SME capturing a picture of the participant's perspective from the rifle mounted camera then annotating the picture in real-time on their workstation. The annotated image, which showed the order of individuals to be tagged, was sent to the participant to initiate the tagging action. The Video with Audio condition consisted of the SME monitoring the participant's perspective while supplying verbal instructions of the individual to be tagged. The Video with Markup and Audio condition combined the Audio and Video conditions so that the SME and participant were able to talk to each other as well as send annotated images.

For each condition, participants tagged unique individuals. They were asked to complete the task as fast as possible without making any errors.

5 Results

Mean task completion time and standard errors for the four experimental conditions are displayed in Figure 9.

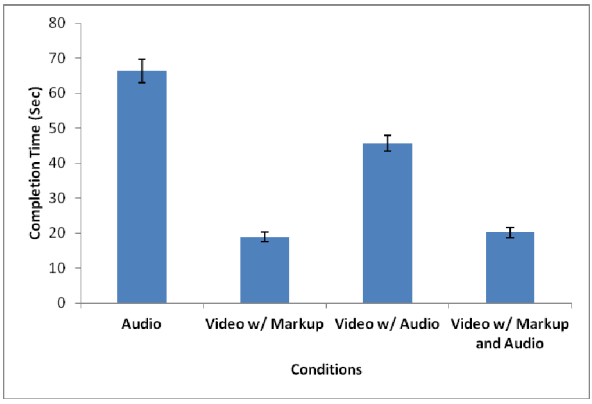


Fig. 9. Mean completion times for each of the four experimental conditions. Error bars are standard errors.

A four condition repeated measures Analysis of Variance (ANOVA) of these data revealed a statistically significant main effect for conditions, F (3,33) = 70.41, $p <$ .05. A subsequent post hoc Tukey-test with alpha set at .05 revealed that participants using Video with Markup and Video with Markup and Audio completed the task statistically faster than the other conditions but were not different from each other. The Tukey-test also found that participants using Video and Audio were statistically faster than Audio alone.

Mean accuracy and standard errors for the four experimental conditions are display in Figure 10.

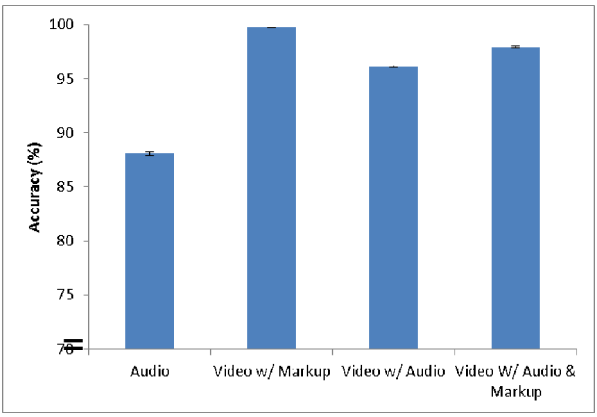


Fig. 10. Mean accuracy for each of the four experimental conditions. Error bars are standard errors.

A four condition repeated measures ANOVA was performed on these data and revealed that the mean accuracy values in the four conditions did not statistically differ from each other, F (3,33) = 2.24, $p > .05$.

Additionally, we examined the degree to which the interfaces provided affected the total verbal communication time. It was found that the style and amount of verbal information relayed between cooperative pairs differed when a shared visual perspective was available. Figure 11 shows the mean voice usage times the remote SME required to achieve common ground positively identifying the experimental targets. A t-test revealed that the Audio condition required more communication time then the Video w/ Audio, t (7) = 4.27, p < .05.

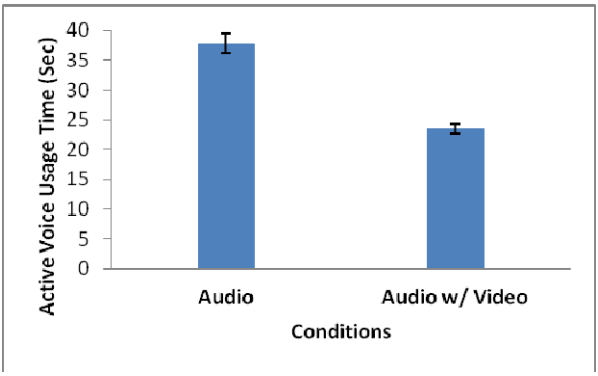


Fig. 11. Comparison of active voice usage time of audio conditions

6 Discussion

Our results are in line with other research evaluating the utility of various multimodal displays in remote collaboration tasks. Our data suggest that multimodal remote

collaboration can be effective on mobile devices, such as smartphones and handheld tablets. Our findings highlight the fact that, when using shared visual perspective, cooperative teams can complete distributed physical tasks quicker and with the same level of accuracy as the other experimental modality conditions.

Not only did having a shared visual perspective result in a faster convergence of understanding, but it also had an impact on the style of the verbal directives. For example, in the Audio only condition, with no shared visuals, the remote SME's verbal directives were descriptive describing the appearance of the individual of interest (i.e., "the first guy has no hat [pause] white beard [pause] and a gray shirt", "the next guy has a brown hat [pause] small black beard [pause] and a white shirt"). Alternatively, in the Video with Audio condition, the verbal directives provided contextual information on the targets' location in the shared visual scene (i.e., "all the way to the back next to the car [pause] that one [pause] yep", "the fifth one to the right"). Moreover, in the Video with Audio condition, the remote SME leveraged pronouns such as "that one", "him", "next one" to convey and direct the worker's aim towards the correct target. The descriptive and contextual information we experienced is similar to the classification of utterance ideas of *Referents* and *Position* presented in Kraut et al [8].

Results also revealed that participants were quicker at completing the task when the helper sent annotated images than when they gave only verbal directions. This could result from the fact that both the helper and participants simply had to look at the image to interpret the directives rather than interpreting the spoken message. This finding supports the famous adage "A picture is worth a thousand words".

7 Future Work

Network/Visual optimization can further enhance our prototype Android application to be executed in bandwidth-limited environments. As we highlighted in the discussion section, collaborating participants preferred to leverage the visual modality to relay directives when given the option. Our current implementation redraws a new 800x600 jpeg still image to the mobile device's display for every helper's markup content. For greater efficiency, we may choose to implement region of interest redraws/image invalidations depicted in Figure 12. This can improve the system in

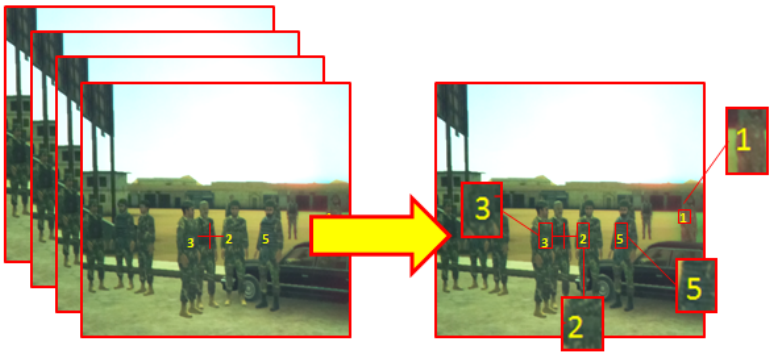


Fig. 12. Region of interest refresh

two ways: 1) It will improve bandwidth efficiency by only transmitting the bounding contexts contained in the markup and 2) It could be used to draw attention to the helper's markup with a distinguishing highlighting color indicating to the worker a change in information.

As the typical use case for military operations is not in a static position or environment, an evaluation of our developmental Android application's performance is warranted when worn in a new location and orientation rather than mounted to a secure object. Recent developments in low-profile wrist mounts (see Figure 13) offer potential test wearable configurations that may be conducive to dismounted on-the-move operations.

Fig. 13. Wrist mounted mobile devices

8 Conclusion

Computer supported cooperative work and team collaboration research has shown that sharing multimodal information (video and audio) enhances cooperative task completion. This paper reported the implementation of an interactive Android collaborative application and an initial evaluation executing a find, fix, and tag scenario. Results from the experiment showed that mobile devices can support the communication capability to successfully complete a task jointly performed by a worker and a remote helper. The cooperative use of annotated images proved to be the most effective means to relay instructional directives. This evaluation provides justification for continual development of mobile, multimodal collaborative applications for distributed operators. Such applications can enhance mission effectiveness by providing BA with a means to more effectively collaborate and coordinate critical military tasks with a remotely located expert.

References

1. Kraut, R.E., Fussell, S.R., Seigel, J.: Visual Information as a Conversational Resource in Collaborative Physical Tasks. Human Computer Interaction 18, 13–49
2. Clark, H., Wilkes-Gibbs, D.: Referring as a Collaborative Process. Cognition 22, 1–39 (1986)
3. Brehmer, B.: Dynamic Decision Making: Human Control of Complex Systems. Acta Psychologies 81(3), 211–241 (1992)
4. Kuzuoka, H., Kosuge, T., Tanaka, K.: GestureCam: A Video Communication System for Sympathetic Remote Collaboration. In: CSCW 1994, pp. 35–45. ACM Press, New York (1994)
5. Fussel, S.R., Kraut, R.E., Siegel, J.: Coordination of Communication: Effects of Shared Visual Context on Collaborative Work. In: CSCW 2000, Philadelphia, PA (2000)
6. Ou, J., Fussell, S.R., Chen, X., Setlock, L.D., Yang, J.: Gestural Communication over Video Stream: Supporting Multimodal Interaction for Remote Colloborative Physical Tasks. In: International Conference on Multimodal Interfaces. ACM Press, Vancouver (2003)
7. Stevenson, D., Li, J., Smith, J., Hutchins, M.: A Collaborative Guidance Case Study. In: AUIC 2008, Wollongong, NSW, Australia (2008)
8. Kraut, R.E., Darren, G., Fussell, S.R.: The Use of Visual Information in Shared Visual Co-Presence. In: CSCW 2002, New Orleans (2002)

Metric Based Automatic Event Segmentation

Yuwen Zhuang[1], Mikhail Belkin[1], and Simon Dennis[2]

[1] Department of Computer Science and Engineering, Ohio State University
Columbus, OH 43201, USA
zhuang.14@buckeyemail.osu.edu, mbelkin@cse.ohio-state.edu
[2] Department of Psychology, Ohio State University
Columbus, OH 43201, USA
simon.dennis@gmail.com

Abstract. This paper describes a metric-based model for event segmentation of sensor data recorded by a mobile phone worn around subjects' necks during their daily life. More specifically, we aim at detecting human daily event boundaries by analysing the recorded triaxial accelerometer signals and images sequence (lifelog data). In the experiments, different signal representations and three boundary detection models are evaluated on a corpus of 2 subjects over total 24 days. The contribution of this paper is three-fold. First, we find that using accelerometer signals can provide much more reliable and significantly better performance than using image signals with MPEG-7 low level features. Second, the models using the accelerometer data based on the world's coordinates system can provide equally or even much better performance than using the accelerometer data based on the device's coordinates system. Finally, our proposed model has a better performance than the state of the art system [1].

Keywords: Triaxial accelerometer, Event segmentation, Lifelog data.

1 Introduction and Motivation

Lifelogging, as a growing interest, is a term referring to people digitally capturing all the information produced by them in daily life. Lifelog is a data collection of records of an individual's daily activities in one or more media forms. This may contain huge volumes of data from different sensor sources. For example, in the study of [2], an average collection of 1,900 images per day per person leads to approximate 700,000 images per year per person. Hence, a main challenge is how to add utility to this huge and complex collection that is continuously captured and accumulated from multiple sensors [3].

Peoples' daily lives consist of different events that come in many varieties. An event is an organization of human experience, such that a dynamic and continuous experience is divided into stable entities, providing a structure for attention, memory and learning [4]. Event segmentation, as one of the most fundamental intelligent mechanisms that a human possess, is a process where people segment a continuous stream of experience into meaningful events [5].

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 129–148, 2013.

Recently, event segmentation [6,1] has been suggested as a method to organize lifelogs where an event boundary occurs when there is an end of one meaningful event and another event begins.

There are several psychologic foundations for using the event segmentation as a methodology for organizing lifelogs. First, supported by behavioral [7] and neuro-imaging data [8,9], it has been suggested that event segmentation is automatic [10,11], even as observers passively view activities. Second, segmenting activity into meaningful events is indicated as a core component of perception [12] and has consequences for memory and learning [13,14]. For example, first, [5] show that individuals who are better able to segment ongoing activity into events are better able to remember it. Second, by event segmentation, a terrific economy of representation for perception and memory can be achieved. Hence, organizing lifelogs into events will provide a more natural and pellucid way for organizing lifelogs, retrieval and interpretion.

However, several related and crucial questions for event segmentation still remain. First, are the event boundaries consistent across different people? In other words, is there good inter-subjective agreement on the event segmentation boundaries? Although the event boundaries can be **fuzzy** [12] indicating there is inevitable variability when and where a boundary occurs, they are **remarkably consistent** across participants [15] with **good reliability** among the same participants in test-retest [7].

Second, what kind of features are needed? [5] show that event boundaries can be identified by tracking significant changes in physical and social features. [12] indicates that the critical features may include sensory features, such as color, sound and movement, and conceptual features, such as cause-and-effect interactions and actors' goals. Evidences [12] are shown to demonstrate that both physical-movement features (such as change in location) and changes in actor's goals play strongly important roles in the segmentation of activity into events.

Finally, since previous studies [10,16,17] only study the movement features on event segmentation for the short time records which last only for a few seconds to several minutes, the final question is - What is the impact for movement features for event segmentation for long time records (which last more than 2 hours or even longer)?

In this paper, our study on natural and realistic daily event segmentation shows that, for daily event segmentation on long time records, the movement features can also play an important role and are important cues for event boundary detection. More specifically, our study focuses on the impact of movement feature and visual feature characterized by two independent sensor sources, namely, accelerometer and image on event segmentation. Furthermore, we will also follow the same way [1] to construct the ground truth boundaries, that is asking people to segment their events by viewing and marking event boundaries on their corresponding image records. But instead of associating the sensor data

to images [1] to evaluate performance, we suggest a performance measure which is well developed from audio segmentation for daily event segmentation using sensor data. This measure considers the "fuzzy" effect [12] of event boundaries by assuming that the event boundary can have a small continuous time interval and evaluates the F-score (a measure of a test's accuracy based on precision and recall) directly on event boundaries. This is different from [1] which evaluates F score on images and [18] which evaluates F score on events (activities).

The specific contributions of this paper are the following:

1. Accelerometer signals can provide much more reliable and significantly better performance than using images signal with MPEG-7 low level features.
2. For the accelerometer signal, our proposed model using the Fourier Transform feature has a better performance than the state of the art system [1] using "the rate of change in motion" ([6,19]) feature for accelerometer signal and also their fusion method.
3. Using the "Behavior Text" [20] feature suggested by Carnegie Mellon University group doesn't give us a better performance than using the traditional Fast Fourier Transform (FFT) feature for event segmentation under our proposed model.

2 Related Work

In one of the early studies, by using a mobile phone equipped with a sensor box, [21] investigated several time series segmentation methods for segmenting context data sequences into discrete, non-overlapping and internally homogeneous segments. A cost function is defined for any segments on the time series. Hence, the segmentation problem has been converted into an optimization problem. In their particular study, they define the segmentation cost as a sum of the variance of the components of the segment. However, there are two drawbacks here. First, they don't have any detailed ground truth boundaries to compare with their methods. In other words, they lack some systematic metric to evaluate the performance. Second, the number of events needs to be predefined and they don't show how to get the optimal number.

An algorithm based on a hidden Markov model (HMM) is proposed by [22] for unsupervised clustering of free-living human activities on accelerometry. This algorithm iteratively trains the HMM whose state is a sub-HMM with minimum duration constraint using the Expectation-Maximisation (EM) algorithm. The topology of the HMM changes during the cluster merging step with a merging criterion. However, this model suffers from two main limitations as mentioned in the paper [22]. First, the varied durations of different activities complicates the selection of features and hyper-parameters. Second, only one hour data sequences have been tested in the experiments instead of a full range of daily human activities.

By using the SenseCam[1], [6] investigated 5 different sources of information, which are low-level image descriptors, audio, temperature, light and accelerometer readings and their combinations to segment the SenseCam images into discrete events. Their method mainly involves two steps. The first step is called score assignment. Generally, the computation of the score involves the distance computation between every pair of contiguous windows which contains a fixed number of data units (e.g. images) and slides along the data sequence ordered by time. In this step, only the time break which has an image timestamp associated with it has a score. A high score will indicate an event boundary. Since different types of data may have different captured times, interpolation techniques are used to get the score for the image capture time. For example, a Gaussian window centred at the capture time of the images is used for the sensor values. For the audio data, a linear interpolation is applied. The second step is about score normalization and threshold. That after score normalization, the segmentation algorithm determines a threshold in order to get 20 events for each day. Generally, the time breaks of the 20 top scorings are considered to be the event boundaries.

In their follow up works, [1] introduces a performance metric by providing ground truth boundaries such that some images are selected as event boundaries. Their performance measure is the F1 score (a measure of a test's accuracy based on precision and recall) between ground truth boundaries and algorithm outputs. In this study, they focus on the segmentation of images in conjunction with accelerometer readings and suggest to use "the rate of change in motion" [6,19] feature for accelerometer data representation.

More recently, [18] propose a framework of the lifelog system by using a smart phone. They investigate the activities segmentation and activities recognition on data collected from 2 users wearing the phone for 5 days. They propose a novel method called "behaviour text" [24,20] to represent the sensory data through quantizing them. In this method, a K-means clustering algorithm is applied to the raw sensor data as the first step. Each sensor record is then assigned a unique symbol sequence presenting its nearest cluster. After converting the raw sensor data into this "behaviour text", they proposed two different methods, namely, top-down activity segmentation through activity change detection for event segmentation and smoothed Hidden Markov Model (HMM) for activities segmentation and annotation. By average over all activity types, the authors find that the top-down activity segmentation approach performs better than the smoothed HMM [18].

3 Smart Phone and Data Collection

Several research groups [25,26,3,20,18] have developed personal Lifelog systems to capture personal experiences by wearing various sensors and a wearable

[1] SenseCam [23] is a small device that people can wear around their neck. It has a digital camera and multiple sensors equipped, including: a light sensor, a thermometer, an accelerometer to detect motion and a passive infra-red sensor to detect the presence of a person.

computer. However, most of them need multiple devices to be carried around user's body. In our study, only a smart phone is needed for data collection which makes it more comfortable for users and encourages more natural interactions with them.

Generally, an Android smart phone contains a variety of sensors including a 3-axis accelerometer, a 3-axis orientation sensor, an light sensor, a magnetic field sensor, a temperature sensor, a pressure sensor or even gyroscope sensor and gravity sensor. It also has the function to take images, videos and record audio. Furthermore, an Android phone can also track the user's location by using a GPS device when the user is outdoors.

In this study, we collect data from 2 subjects who use an Android phone to capture images, audio, GPS locations and some sensor data as they engage in their every day activities. The phone is worn around their neck and is positioned in a case with a strap as shown in Figure 1. They are free to turn the application off when they want to protect their privacy. However, they are instructed to provide at least 6-7 hours worth of data each day.

Fig. 1. Smart phone and software for collecting activity data

4 Metric-Based Event Segmentation

Generally, this model involves three steps, namely, "Feature Extraction", "Distance Metrics" and "Event Boundary Detection". The overall procedure for sensor data is depicted in Figure 2, the procedure for the image data is similar.

4.1 Feature Extraction

Sensor Data

1. Fast Fourier Transform (FFT) feature: The signal is first divided into a series of consecutive overlapping frames where each frame is a fragment of the signal - a fixed size of samples. At a sampling frequency of 15HZ, a sample

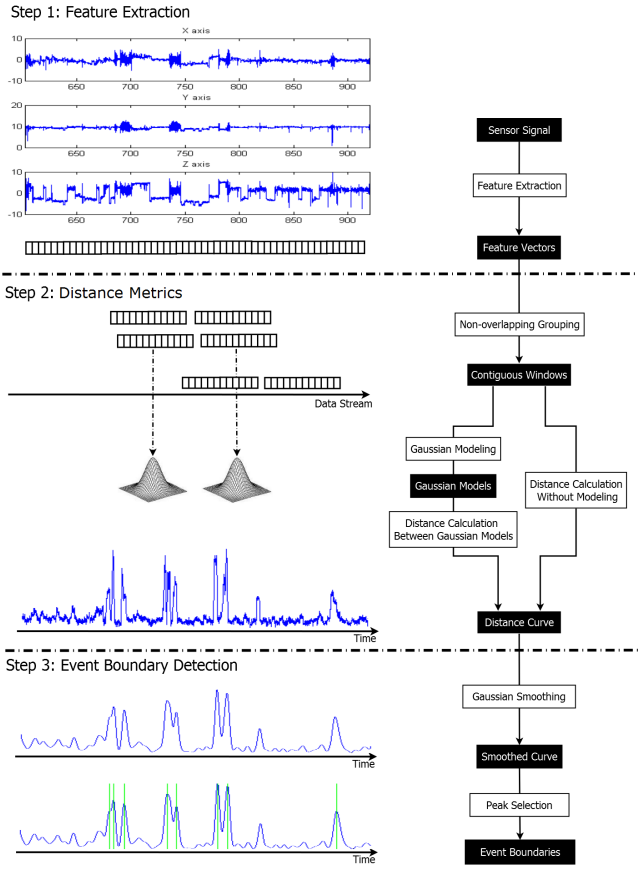


Fig. 2. Metric-based Event Segmentation for Sensor Data

size of 30 samples represents 2 seconds. FFT features are then extracted for each frame. In the experiment, the sample size is empirically set to 30 and the overlapping size is 10.

Suppose x is the input signal and y is the output features, the function implementing this transform is as follows,

$$y_k = \sum_{j=1}^{N} x_j \omega_N^{(j-1)(k-1)}$$

where $\omega_N = e^{(-2\pi i)/N}$ is an Nth root of unity and the length of x is N. And

$$Y_k = |y_k|$$

where $|\cdot|$ is the complex magnitude.

Since $Y_{\frac{n}{2}-k} = Y_{\frac{n}{2}+k}$, we can just keep half of the FFT features. Furthermore, the DC feature (the first one Y_1) is the total acceleration value of the signal over the window and is discarded in this study since from the experiment results we find excluding it can achieve a better performance. This (Excluding the DC component) is similar to how [27] handles activity recognition. Hence, the selected FFT features for each frame are $\{Y_k\}$ where $k \in \{2, 3, 4, \cdots, \frac{n}{2}\}$. Finally, since the accelerometer has three axis, the final feature vector is the concatenation of the FFT features for each axis.
2. Motion Change Rate (MCR) feature: "The rate of change of motion" is first introduced by [19] and suggested by [6,1] for accelerometer data captured by SenseCam on event segmentation.
3. "Behaviour Text" feature: This feature is suggested by [24,20,18] to represent the sensory data for event segmentation and activity recognition. A k-means clustering method is applied on the raw sensor data that results a sequence of symbols representing the cluster centres for each sample.

Image Data. The images are represented by MPEG-7 descriptors. The descriptors we select are following what [1] suggest which are: color layout, color structure, scalable color and edge histogram.

4.2 Distance Metrics

The distance metrics results in a curve of dissimilarity called "distance curve" with respect to time.

Sensor Data

1. For Fast Fourier Transform (FFT) feature: Feature vectors are grouped into a series of non-overlapping consecutive windows (sliding windows) whose size is fixed. A dissimilarity measure is then applied on pairwise sliding windows. The step length is one frame which means after the dissimilarity measure on two non-overlapping consecutive windows, we move both windows one frame in the direction of increasing time and compute the new dissimilarity and so on. In the experiment, we test different window sizes and show their corresponding performances.
   - For the **model-based** approach, a multivariate Gaussian distribution is applied for each window to describe the data. Several distance measurements between these two Gaussians are used to measure the dissimilarity of neighboring non-overlapping windows and the windows are then shifted by a fixed step (about 1 frame) along the whole signal. This process leads to the final distance curve. The following are several optional distance measurements:
     Kullback-Leibler diatance **(KL)**:

$$d_{KL} = \frac{1}{2}(\mu_1 - \mu_2)^T(\Sigma_1^{-1} + \Sigma_2^{-1})(\mu_1 - \mu_2) + \frac{1}{2}tr(\Sigma_1^{-1}\Sigma_2 + \Sigma_2^{-1}\Sigma_1 - 2I) \quad (1)$$

Bhattacharyya distance **(BHA)**:

$$d_{BHA} = \frac{1}{4}(\mu_1 - \mu_2)^T(\Sigma_1 + \Sigma_2)^{-1}(\mu_1 - \mu_2) + \frac{1}{2}log\frac{|\Sigma_1 + \Sigma_2|}{2\sqrt{|\Sigma_1||\Sigma_2|}} \quad (2)$$

Bayesian Information Criterion **(BIC)**[28]:

$$d_{BIC} = Nlog|\Sigma| - N_1log|\Sigma_1| - N_2log|\Sigma_2| - \lambda P \quad (3)$$

where Σ is the sample covariance matrix from samples of two windows and $P = \frac{1}{2}(d + \frac{1}{2}d(d+1))logN$. d is the dimension of the space, the penalty weight λ is equal to 1, N is the total number of samples in two windows and μ is the mean vector and I is identity matrix.

– For the **non-model-based** approach, we introduce the following method: Average Euclidean Distance **(AED)**:

$$d_{AED} = \frac{1}{|A||B|}\sum_{u\in A, v\in B} dist(u, v) \quad (4)$$

where $dist(u, v)$ is a Euclidean distance between vector u and vector v. Set A and B represent two sliding windows and $|\cdot|$ denotes the cardinality. Mean Vector Distance **(MVD)**:

$$d_{MVD} = dist(\mu_1, \mu_2) \quad (5)$$

where $dist(\mu_1, \mu_2)$ is a Euclidean distance between the mean vector μ_1 and μ_2 for sliding windows.

2. For the Motion Change Rate (MCR) feature: In order to compare the system performance suggested by [6,1], the sensor motion values are associated with an image using a Gaussian window centred at the time the image is captured. The distance curve is then formed from a series of motion values where large motion values indicate event boundaries. A Min-Max normalisation technique [29] is applied followed by the event boundary detection [1]. In the experiment, we test different Gaussian window widths and show their corresponding performances.
3. For the "Behaviour Text" feature: Behaviour text string is grouped into a series of non-overlapping consecutive windows (sliding windows) whose size is fixed. A dissimilarity measure [20,18] is then applied on pairwise sliding windows. In the experiment, we test different window sizes and show their corresponding performances.

Image Data. Suggested by [1], images are grouped into a series of non-overlapping consecutive windows (sliding windows) whose size is fixed. An "average image representation" is derived for each window, histogram intersection is then applied on pairwise "average image representation". Since the histogram intersection results a similarity, the dissimilarity is derived by subtracting the similarity from 1. In the experiment, we test different window sizes (different number of images) and show their corresponding performances.

4.3 Event Boundary Detection

This step involves two sub steps, namely "Smoothing" and "Peak Selection".

Smoothing. Since the event boundaries are "fuzzy"[12] and the distance curve may contain noise and events may have different levels of granularity or scale, it is necessary to smooth the curve to identify robust event boundaries that are invariant with respect to granularity or scaling, and are minimally affected by noise and small distortions. Suggested by [30], Gaussian kernel is used to handle these considerations. The following is the description:

To smooth the curve by the convolution of a variable-scale Gaussian, $G(t, \sigma)$, with an input curve, $I(t)$:

$$L(t, \sigma) = G(t, \sigma) * I(t), \tag{6}$$

where $*$ is the convolution operation in t, and

$$G(t, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{t^2}{2\sigma^2}} \tag{7}$$

The smoothing level for Gaussian kernel is denoted by parameter σ. In the experiment, the σ for cross validation is from $\{1, 2, 3, ..., 90\}$.

Peak Selection. Three different peak selection models are proposed in this section, namely, "All peak", "Tall peak" and "Significant peak". In the experiments, their corresponding free parameters are selected by cross validations.

Firstly, for the "All peak", this model simply selects all the peaks (potentional boundaries) to form the final event boundaries. Since peaks can be selected from different smoothing levels, there is only one free parameter in this model - the smoothing level σ in "Gaussian Convolution".

Secondly, a peak is "tall" when its height is greater than some *threshold* as Figure 3 depicts. In this model, all the boundaries corresponding to tall peaks are selected as the final event boundaries. In this model, besides the σ in "Gaussian Convolution", there is another free parameter - the *threshold*. In the experiment, the σ we select for cross validation is from $\{1, 2, 3, ..., 90\}$ and the *threshold* is $i \times \frac{(max-min)}{50} + min$ where $i \in \{1, 2, 3, ..., 50\}$ and max and min are the maximum and minimum values for all the distance curves.

Finally, a peak is "significant"[31] if

$$\begin{aligned} |d(max) - d(min_{left})| &> \alpha\sigma' \\ or\ |d(max) - d(min_{right})| &> \alpha\sigma' \end{aligned} \tag{8}$$

where α is a parameter, σ' is the standard deviation of the distance curve. And min_{left} and min_{right} are the left and right minimas around the peak "max" as Figure 4 depicts.

This model selects all the "significant peak". The boundaries associated to these peaks are selected to form the final event boundaries. This model involves two free parameters, σ in "Gaussian Convolution" and α in detecting the "significant peak". In the experiment, the α we selected for cross validation is from $\{0.1, 0.2, 0.3, ..., 2.0\}$ and the σ for cross validation is from $\{1, 2, 3, ..., 90\}$.

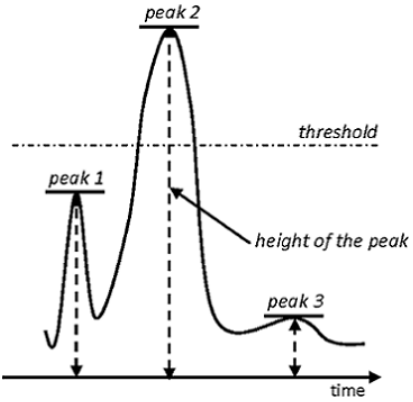


Fig. 3. Tall Peak: Peak 2 is tall peak since its height is greater than the threshold

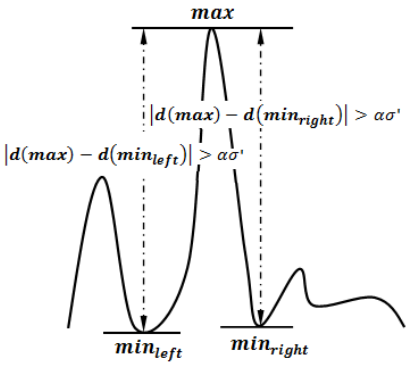


Fig. 4. Significant Peak: The middle peak is "Significant"

5 Segmentation Quality Measure

5.1 Evaluation Reference

In order to evaluate performance, subjects manually segment their daily experience into different events by viewing their corresponding sequential images. The timestamps of the images they choose are marked as ground truth boundaries. Hence, the ground truth event boundaries for daily experience is a set of timestamps.

5.2 Evaluation Metrics

An event segmentation system tries to detect changes in the sensor signals where the changes correspond to boundaries of different events. Such a system may have two possible types of error. $Type-1$ errors occur if a true boundary is not hit within a certain range in time (3 minutes either side in our case). $Type-2$ errors occur if a detected change does not correspond to any ground truth boundary (false alarm). Type 1 and 2 errors are also referred to as precision (PRC) and recall (RCL), respectively [32].

Let N_{hit} be the number of boundaries correctly detected (hit), N_{ref} be the total number of boundaries in the reference and N_f be the number of detected boundaries (system outputs).

The precision (PRC) and recall (RCL) can be defined as follows:

$$PRC = \frac{N_{hit}}{N_f}, RCL = \frac{N_{hit}}{N_{ref}} \tag{9}$$

Generally, the F-measure is often used to compare the performance of different algorithms as follows:

$$F = \frac{2 \times PRC \times RCL}{PRC + RCL} \tag{10}$$

The range of the F-measure is from 0 to 1 where a higher F-measure indicates a better performance.

5.3 Hits Counting: Search Region

In order to determine the number of hits, a fixed-size search region around each reference boundary is placed and verified whether there are some boundaries produced by a segmentation algorithm in these regions as Figure 5 shows.

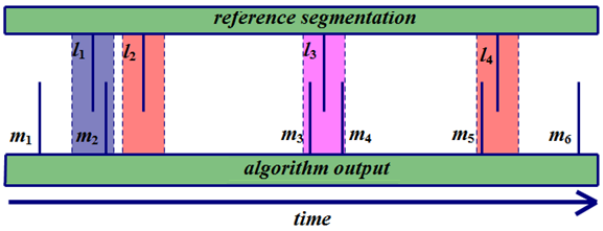


Fig. 5. In this example, for reference boundary l_1, there is an algorithm output m_2 in its search range (colored region with dotted lines as its both sides), hence, boundary l_1 is hit by output m_2. For boundary l_3, there are two outputs in its search range, and l_3 is also hit. The number of boundaries correctly detected is 3 in this case (N_{hit}=3). And the total number of boundaries in the reference is 4 (N_{ref}=4). Total number of detected boundaries (algorithm outputs) is 6 (N_f=6).

However, if there is some overlapping search region, this will cause an ambiguous situation in evaluation. This problem can be solved by removing the overlapping region by asymmetrically shrinking the search regions of its two sides to a common mid-point [33] (see Figure 6).

In the experiments, since we don't study segmentation of the events which last less than 3 minutes, the total search region is set to 6 minutes.

6 Experiments

6.1 Dataset

Sensor Data. Unlike [6] whose sensor data is captured every 2 seconds (0.5HZ), the sampling rate of our smart phone sensor is from 15HZ to 20HZ. This sample rate is sufficient for detecting human daily physical activity [34]. Since the **raw accelerometer** data is recorded using the **device's coordinate system**, we convert it into **world's coordinate system** (see Figure 7) by eliminating the force of the gravity and with the assistance of the magnetometer sensor. We call this converted data (which is also gravity eliminated) **adjusted accelerometer** data. In the experiments, we investigate the performance difference between these two different representations by using different distance measures and different models.

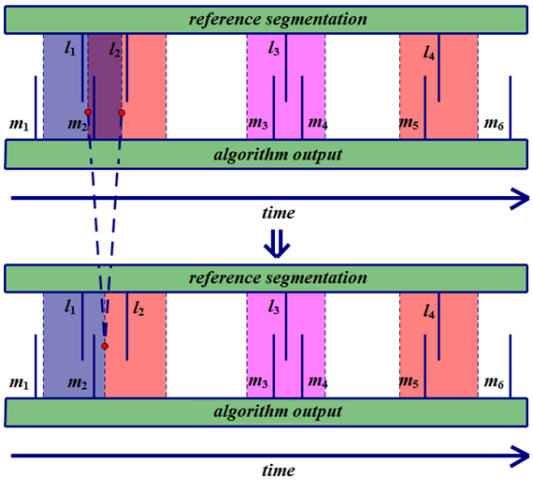


Fig. 6. The first plot shows an example of an overlapping search region causing a ambiguous situation in evaluation, namely a problem of how to define a matching boundary for each reference boundary. It is not clear whether l_1 or l_2 is hit by m_2 or both. The second plot removes the overlapping by asymmetrically shrinking the search regions of its two sides to a common mid-point. Hence, the matching becomes straightforward so that m_2 is only contributed to the hit of l_1.

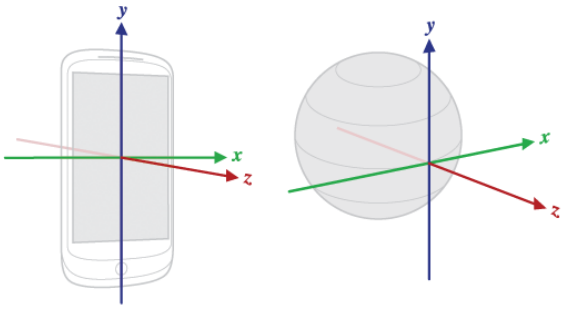


Fig. 7. The left figure indicates the device's coordinate system and the right figure represents the world's coordinate

Ground Truth Boundaries. In this experiment, we ask the wearers to manually create a ground truth of segmentations for all his/her own recorded images. There are several reasons that we ask the wearers to segment their own records. First, event segmentation has subjective and individual differences and is hard to characterize by normative criteria [9]. Second, we believe that the wearers have the best knowledge of their own intentions and goals for what they did when viewing their own image records. Third, considering the privacy issues with data highly personal to the user, it is desirable to mark event boundaries according to each wearers' own judgements suggested by [1].

6.2 Experimental Setup

In the experiments, the data set is first grouped into continuous chunks in time. Then all the chunks are divided into training chunks and test chunks. Each chunk represents records of activities per person per day. We leave one chunk out to select the parameters and test it on the test chunk. By grouping the data into chunks during the testing, this can guarantee that the models are tested on completely different days.

6.3 Understanding of the Performance Measure

In order to have a better intuition of the performance measure - F values, we do some experiments based on random boundary generation and ground truth boundary replacement.

Random Boundary Generation. Without knowing the number of event boundaries for each chunk, we use a fixed number to generate the random boundaries from a uniform distribution. Fig 8 shows the results, the number of boundaries range from 1 to 200 for all chunks. The F values come from the average results on 100 simulations (random boundary generation). Furthermore, knowing the number of event boundaries for each chunk, we can get a F value that is 0.22 ± 0.026, where 0.22 is the average and 0.026 is the standard deviation. The average suggests a "chance segmentation" and can be counted as a baseline.

Ground Truth Boundary Replacement. In this experiment, different proportion of ground truth boundaries are replaced by equal number of random boundaries with 100 different simulations. Fig 9 shows the results. For example, a F value of 0.65 indicates that nearly 38% of the ground truth boundaries are placed by random boundaries. 0.60 indicates 44%, 0.55 indicates 51% and 0.5 indicates 59%.

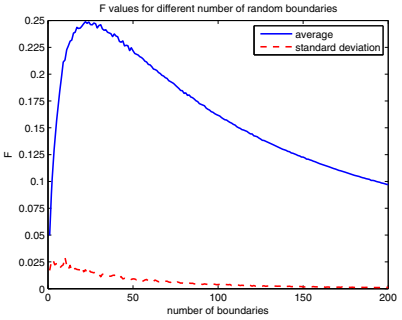


Fig. 8. F values for different number of boundaries

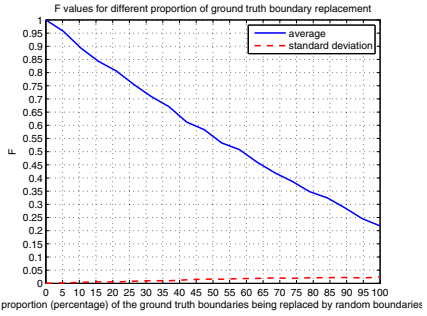


Fig. 9. F values for different proportion of ground truth boundary replacement

6.4 Experimental Results and Analysis

In the experiments, we are going to answer the follows: 1. How well can our proposed model do comparing with other systems on the event segmentation? 2. Which modality is better? Image data or accelerometer data? 3. What is the impact for movement features for event segmentation on long time records?

1. Our Proposed Models: Fig 10 and Fig 11 show the results. By using the "Tall Peak Detection" on adjusted accelerometer, the "BHA" and "BIC" distance measures can give us the best average F value around 0.65 using the window size in the range from 50 frames (equivalent to 75 seconds) to 90 frames (equaling 135 seconds) [2].

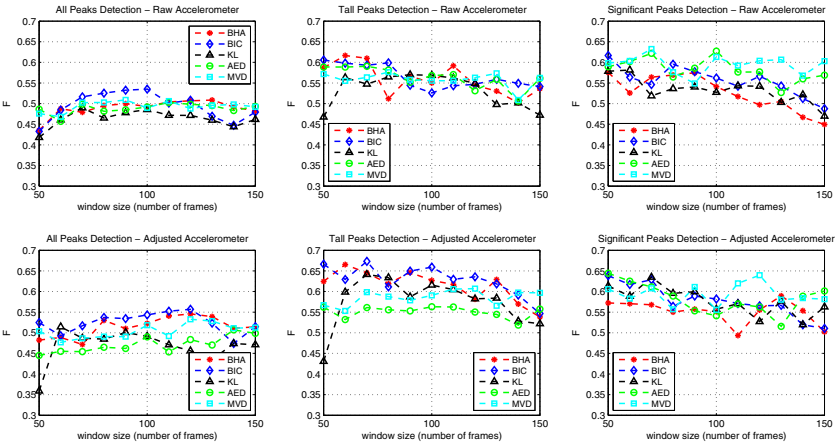


Fig. 10. Event segmentation on accelerometer data using our proposed model

The "Histogram Intersection" distance suggested by [35] is used in the distance curve computation for image signals with different representations. By the comparison of the event segmentation results between using accelerometer data (Fig 10) and image data (Fig 11), it is clear, the event segmentation performance from using accelerometer data is much higher than using image data.

2. Dublin City University's System: The Motion Change Rate (MCR) feature is used in this system. Since the performance measures are different, in order to make a fair comparison, the distance curves are processed with/without "Peak Scoring" technique before peak detection. Furthermore, a data normalisation method "Sum" is used for different signals before fusion. A fusion

[2] Although different models would prefer different window sizes, we would like to report the average F value in a reasonably good range instead of reporting the best one. Since the best one associated with a particular window size may be sensitive to the data and model.

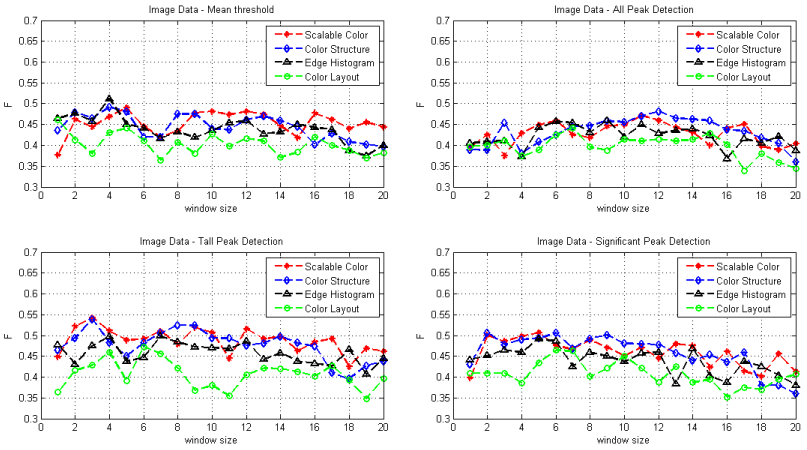


Fig. 11. Event segmentation on image data using our proposed model

method "CombMIN" suggested by [1] is used for different signals. For this fusion method, different distance curves are multiplied by different weights. For each time step, the minimal among these weighted distance curves is selected as the output. The weights we used in the fusion are suggested by [35].

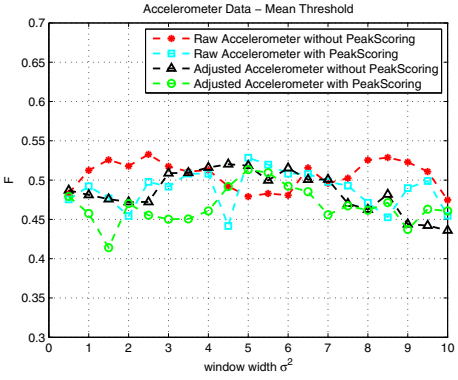


Fig. 12. Event segmentation using Dublin City University's System - Accelerometer Data Only

Fig 12 shows the results for accelerometer data using the "Mean Threshold" method suggested by [1] for peak detection. The best performance using the accelerometer data is around 0.53.

The early fusion method is suggested by [35] (left panel in Fig 13) so that different image representations are concatenated into a signal representation,

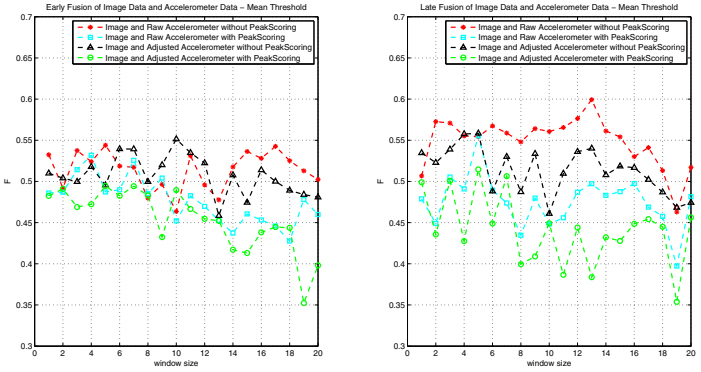


Fig. 13. Event segmentation using Dublin City University's System - Fusion

then a distance curve computed on that representation is fused with distance curve of the accelerometer signal later. The weight for the image distance curve is 0.65 and the weight for the accelerometer distance curve is 0.35. Furthermore, we suggested a completely late fusion method (right panel in Fig 13) where this fusion is on distance curves of different image representations and distance curves of accelerometer signal with equal weights. And it is clear, when using raw accelerometer data without "peak scoring", this method can achieve an F score which is around 0.58. Finally, with a comparison between Fig 13 and Fig 12, the fusion of image and accelerometer data can give us a better result than using accelerometer data only.

3. Carnegie Mellon University's system: The "Behaviour Text" feature is used in this system. In this experiment, in order to make a fair comparison[3] for the study of the "Behaviour Text" feature for event segmentation, we use our proposed peak detection methods to find the event boundaries. The hyper parameters for the feature extraction and dissimilarity measure are set empirically [24,20] according to the experimental results. Fig 14 shows the results. Using "Significant Peak Detection" on adjusted accelerometer data, the best average F values are around 0.60 whose window sizes are from 1 minute length to 2 minute length. And it is clear that the performances using adjusted accelerometer data are much better than the performances using raw accelerometer data.

[3] The papers [24,20] lack enough details for replicating their hierarchical segmentation method and through some email contacts with the main author, our attempt for replicating their hierarchical segmentation method still gives us some low performance.

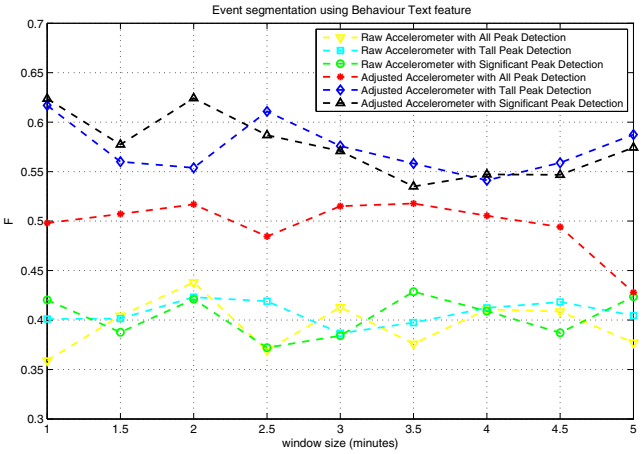


Fig. 14. Event segmentation using "Behaviour Text" feature

7 Conclusion and Discussion

The automatic event segmentation using accelerometer data appears to be more reliable than that using image data. This was somewhat unexpected considering the ground-truth boundaries were created by subjects from imagery. The main reason may be due to the fact that the camera can capture a totally different image signal with slightly different device orientation even within a similar context. Humans can understand the semantic meaning of these images and recognize that they have a similar context even though they may look very different. However, the distance measure using MPEG-7 low level features will tell the difference in vision and the metric based model with these features fails to understand the similarity in context.

Second, according to the comparison between our proposed model and the Dublin City University System [1] on event segmentation using accelerometer data, we find it is necessary to have a high sampling rate to capture the change of daily human activity. One reason that our proposed model can have a higher performance than the Dublin City University System may be due to their low sampling rate - sensor data is captured every 2 seconds (0.5HZ). We think such low sampling rate may be insufficient for detecting daily human physical activity [34] and may fail to detect some event boundaries.

Third, our proposed peak selection methods using the bag of feature representation such as "Behavior Text" [20] suggested by Carnegie Mellon University group has a much better performance on the adjusted accelerometer data than the raw accelerometer data. Furthermore, the best performance from our proposed model is also from the adjusted accelerometer data. All these may suggest that, for the accelerometer data, the gravity impact combined with the orientation of the phone (implied by the accelerometer data based on a device's local coordinate system) is useless or even harmful for event segmentation.

Since the gravity can be decomposed as adding numbers into the three axis of the accelerometer data, they vary with different phone's positions. This usually overwhelm the acceleration introduced by activity change and make what the algorithm detected majorly becomes the change of the device's position.

Finally, we believe that the movement features can play an important role for event segmentation on the long time records.

Acknowledgment. This work was supported by the Air Force Office of Scientific Research under AFOSR FA9550-09-1-0614 and National Science Foundation under IIS-1117707.

References

1. Doherty, A.R., Smeaton, A.F.: Automatically Segmenting Lifelog Data into Events. In: WIAMIS 2008 - 9th International Workshop on Image Analysis for Multimedia Interactive Services (2008)
2. Doherty, A.R., Byrne, D., Smeaton, A.F., Jones, G.J.F., Hughes, M.: Investigating Keyframe Selection Methods in the Novel Domain of Passively Captured Visual Lifelogs. In: CIVR 2008: Proceedings of the 2008 International Conference on Content-based Image and Video Retrieval, Niagara Falls, pp. 259–268. ACM (2008)
3. Takata, K., Ma, J., Apduhan, B.O., Huang, R., Jin, Q.: Modeling and Analyzing Individuals Daily Activities Using Lifelog. In: Second International Conference on Embedded Software and Systems, pp. 503–510. IEEE Computer Society, Los Alamitos (2008)
4. Swallow, K.M., Zacks, J.M., Abrams, R.A.: Event Boundaries in Perception Affect Memory Encoding and Updating. Journal of Experimental Psychology. General 138(2), 236–257 (2009)
5. Zacks, J.M., Swallow, K.M.: Event Segmentation. Current Directions in Psychological Science 16(2), 80–84 (2007)
6. Doherty, A.R., Smeaton, A.F., Lee, K., Ellis, D.P.W.: Multimodal Segmentation of Lifelog Data. In: Proc. of RIAO 2007 (2007)
7. Nicole, J.M.Z., Speer, K., Swallow, K.M.: Activation of Human Motion Processing Areas During Event Perception. Cogn. Affect. Behav. Neurosci. 3, 335–345 (2003)
8. Chen, C.: Information Visualisation and Virtual Environments. Springer, London (1999)
9. Zacks, J.M., Braver, T.S., Sheridan, M.A., Donaldson, D.I., Snyder, A.Z., Ollinger, J.M., Buckner, R.L., Raichle, M.E.: Human Brain Activity Time-locked to Perceptual Event Boundaries. Nat. Neurosci. 4(6), 651–655 (2001)
10. Zacks, J., Tversky, B., Iyer, G.: Perceiving, Remembering, and Communicating Structure in Events. Journal of Experimental Psychology: General 130, 29–58 (2001)
11. Zacks, J.M., Swallow, K.M., Vettel, J.M., McAvoy, M.P.: Visual Motion and the Neural Correlates of Event Perception. Brain Research 1076(1), 150–162 (2006)
12. Zacks, J.M., Speer, N.K., Swallow, K.M., Braver, T.S., Reynolds, J.R.: Event perception: A mind-brain perspective. Psychological Bulletin 133, 273–293 (2007)
13. Zacks, J.M., Speer, N.K., Vettel, J.M., Jacoby, L.L.: Event Understanding and Memory in Healthy Aging and Dementia of the Alzheimer Type. Psychology and Aging 21(3), 466–482 (2006)

14. Newtson, D., Engquist, G.: Foundations of attribution: The Perceptual Organization of Ongoing Behavior. Journal of Experimental Social Psychology 12(5), 436–450 (1976)
15. Newtson, D.: Foundations of attribution: The perception of ongoing behavior. New Directions in Attribution Research, 223–248 (1976)
16. Zacks, J.M., Kumar, S., Abrams, R.A., Mehta, R.: Using Movement and Intentions to Understand Human Activity. Cognition 112(2), 201–216 (2009)
17. Newtson, D., Engquist, G.A., Bois, J.: The Objective Basis of Behavior Units. Journal of Personality and Social Psychology 35(12), 847–862 (1977)
18. Wu, P., Peng, H.-K., Zhu, J., Zhang, Y.: SensCare: Semi-automatic Activity Summarization System for Elderly Care. In: Zhang, J.Y., Wilkiewicz, J., Nahapetian, A. (eds.) MobiCASE 2011. LNICST, vol. 95, pp. 1–19. Springer, Heidelberg (2012)
19. ÓConaire, C., O'Connor, N.E., Smeaton, A.F., Jones, G.J.F.: Organising a Daily Visual Diary Using Multifeature Clustering. Optical Society of America (2007)
20. Chennuru, S., Chen, P.-W., Zhu, J., Zhang, J.Y.: Mobile Lifelogger – Recording, Indexing, and Understanding a Mobile User's Life. In: Gris, M., Yang, G. (eds.) MobiCASE 2010. LNICST, vol. 76, pp. 263–281. Springer, Heidelberg (2012)
21. Himberg, J., Korpiaho, K., Mannila, H., Tikanmäki, J., Toivonen, H.T.: Time Series Segmentation for Context Recognition in Mobile Devices, pp. 203–210 (2001)
22. Nguyen, A., Moore, D., McCowan, I.: Unsupervised Clustering of Free-living Human Activities Using Ambulatory Accelerometry. In: 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS 2007, pp. 4895–4898. IEEE (2007)
23. Hodges, S., Williams, L., Berry, E., Izadi, S., Srinivasan, J., Butler, A., Smyth, G., Kapur, N., Wood, K.: SenseCam: A Retrospective Memory Aid. In: Dourish, P., Friday, A. (eds.) UbiComp 2006. LNCS, vol. 4206, pp. 177–193. Springer, Heidelberg (2006)
24. Chen, P., Chennuru, S., Zhang, Y.: A Language Approach to Modeling Human Behavior. In: Proc. Seventh Intl. Conf. Language Resources and Evaluation (2010)
25. Hori, T., Aizawa, K.: Context-based Video Retrieval System for the Life-log Applications. In: Proceedings of the 5th ACM SIGMM International Workshop on Multimedia Information Retrieval, MIR 2003, New York, NY, USA, pp. 31–38 (2003)
26. Kim, I.-J., Ahn, S.C., Ko, H., Kim, H.-G.: Automatic Lifelog Media Annotation based on Heterogeneous Sensor Fusion. In: IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, MFI 2008, pp. 703–708 (August 2008)
27. Bao, L., Intille, S.S.: Activity Recognition from User-annotated Acceleration Data, pp. 1–17. Springer (2004)
28. Chen, S.S., Gopalakrishnan, P.S.: Speaker, Environment and Channel Change Detection and Clustering via the Bayesian Information Criterion, pp. 127–132 (1998)
29. Montague, M., Aslam, J.: Relevance Score Normalization for Metasearch. In: Proceedings of the Tenth International Conference on Information and Knowledge Management, pp. 427–433. ACM (2001)
30. Lindeberg, T.: Scale-space theory: A Basic Tool for Analysing Structures at Different Scales. Journal of Applied Statistics, 224–270 (1994)
31. Zochová, P., Radová, V.: Modified Distbic Algorithm for Speaker Change Detection. In: Eurospeech, pp 1:3073–1:3076 (2005)
32. Ajmera, J., Mccowan, I., Bourlard, H.: Robust Speaker Change Detection. IEEE Signal Process. Lett. 11, 649–651 (2004)

33. Räsänen, O.J., Laine, U.K., Altosaar, T.: An Improved Speech Segmentation Quality Measure: the R-Value. In: INTERSPEECH, pp. 1851–1854 (September 2009)
34. Bouten, C., Koekkoek, K., Verduin, M., Kodde, R., Janssen, J.: A Triaxial Accelerometer and Portable Data Processing Unit for the Assessment of Daily Physical Activity. IEEE Transactions on Biomedical Engineering 44(3), 136–147 (1997)
35. Doherty, A.: Providing Effective Memory Retrieval Cues through Automatic Structuring and Augmentation of a Lifelog of Images (2008)

Visage: A Face Interpretation Engine for Smartphone Applications

Xiaochao Yang[1], Chuang-Wen You[1], Hong Lu[2], Mu Lin[1], Nicholas D. Lane[3], and Andrew T. Campbell[1]

[1] Dartmouth College, 6211 Sudikoff Lab, Hanover, NH 03755, USA
[2] Intel Lab, 2200 Mission College Blvd, Santa Clara, CA 95054, USA
[3] Microsoft Research Asia, No. 5 Dan Ling St., Haidian District, Beijing, China
{Xiaochao.Yang,chuang-wen.you}@dartmouth.edu, hong.lu@intel.com, mu.lin@dartmouth.edu, niclane@microsoft.com, campbell@cs.dartmouth.edu

Abstract. Smartphones represent powerful mobile computing devices enabling a wide variety of new applications and opportunities for human interaction, sensing and communications. Because smartphones come with front-facing cameras, it is now possible for users to interact and drive applications based on their facial responses to enable participatory and opportunistic face-aware applications. This paper presents the design, implementation and evaluation of a robust, real-time face interpretation engine for smartphones, called *Visage*, that enables a new class of face-aware applications for smartphones. Visage fuses data streams from the phone's front-facing camera and built-in motion sensors to infer, in an energy-efficient manner, the user's 3D head poses (i.e., the pitch, roll and yaw of user's heads with respect to the phone) and facial expressions (e.g., happy, sad, angry, etc.). Visage supports a set of novel sensing, tracking, and machine learning algorithms on the phone, which are specifically designed to deal with challenges presented by user mobility, varying phone contexts, and resource limitations. Results demonstrate that Visage is effective in different real-world scenarios. Furthermore, we developed two distinct proof-of-concept applications, Streetview+ and Mood Profiler driven by Visage.

Keywords: face-aware mobile application, face interpretation engine.

1 Introduction

Smartphones come with an increasing number of embedded sensors (e.g., microphone, accelerometer, gyroscope, etc.) that are enabling a new generation of sensing applications on the phones, such as, understanding user behaviors [21], life patterns, wellness, and environmental impact [22]. Similar to the microphone [17], the phone camera is also a ubiquitous sensor, which is typically used for recreational purposes such as taking pictures and videos. To date, the camera has not been exploited as a powerful sensing modality in its own right. This is about to change for the following three reasons. First, mobile phones have increasing sensing, computing, storage and communication capabilities. Next, mobile phones have evolved into a multimedia platform capable of downloading a

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 149–168, 2013.

wide variety of applications from Google Play and the App Store. As a result, the way in which users interact with their phones has drastically changed over the past several years; that is, users are increasingly interacting with mobile applications (e.g., tweeting, web surfing, texting) directly through their phones' touch screens. Third, front-facing cameras are becoming a standard feature of new smartphones, allowing for new forms of interactions between the user and the phone, for example, the phone is capable of *observing* users as they interact with different mobile applications in everyday uses. A key contribution of Visage is that it supports a set of sensing, tracking and machine learning algorithms on smartphones, which are specifically designed to deal with the difficult challenges presented by user mobility, varying phone contexts and resource-limited phones (e.g., limited battery lifetime, computational resources).

Traditional human-computer interactions rely on buttons and clicks, which emphasize the explicit communication but neglect the implicit reactions from users. In the 1970s, Ekman [11] found that human emotions express distinctive content together with a unique facial expression. He found that there are 6 prototypical emotion categories that are universal across human ethnicities and cultures, namely happiness, sadness, fear, disgust, surprise and anger. To enable natural ways to interact with applications, previous researchers [30] focus on vision techniques that use fixed cameras (e.g., desktop cameras). While there is a considerable amount of work in computer vision on topics such as face tracking and expression classification, there has been very little work done on solving these problems on mobile phones. On the contrary, shifting from fixed indoor settings to mobile environments, our proposed system specifically deals with varying phone context and unpredictable phone movement. Since mobile phones (equipped with sophisticated sensing, computing, and communication capabilities) have become an indispensable part of our everyday lives, this creates an opportunity to develop a face interpretation engine that senses users' visual feedback and utilizes their face responses to develop new intuitive ways to interact with applications on mobile phones. For example, a user's head rotation angle can be used to infer what the user is paying attention to, such as, points of interest in the real world. The rotation angle information, for example, can be used to drive exploration of 360-degree Google StreetView panoramas.

Most smartphones are equipped with a front-facing camera, which is capable of capturing users' faces and expressions while they are interacting with their phones. In this paper, we present *Visage*, a robust, real-time face interpretation engine for smartphones that enables a new class of face-aware applications on the phone. Visage fuses data streams from the phone's front-facing camera and built-in motion sensors to infer, in an energy-efficient manner, the users' 3D head poses (i.e., the angle of pitch, roll and yaw of the user's head with respect to the phone's orientation) and facial expressions (e.g., happy, sad, angry, etc.). We explore how to make use of this new channel of information in a ubiquitous and nonintrusive way through the implementation of two proof-of-concept face-aware applications based on the Visage engine: 1) StreetView+, which exploits user's head angle to drive exploration of 360-degree StreetView panoramas from

the existing Google Map service; and 2) MoodProfiler, which opportunistically senses users' expressions and provides visualized summary of their moods as they use specific applications (e.g., email client and YouTube viewer) in their daily lives.

The contributions of this paper are three-fold:

- *Context-aware system framework*: We design new sensing, preprocessing, tracking and learning algorithms for efficient inference of users' facial expressions under varying mobility conditions. The Visage face detection algorithm uses gravity estimation provided by the built-in motion sensors to compensate misalignment of the phone's camera and user's face. The system detects phone shakiness to reduce noise and accumulated error to improve head tracking robustness. The Visage preprocessing algorithm uses the light information of the face region to automatically adjust the camera's exposure level to ensure sufficient image quality for tracking and inference algorithms.

- *Resource-aware management*: We study and identify the optimal resolution of the video stream from the front-facing camera to reduce the computation load on the phone while maintaining acceptable real-time inference accuracy. We adaptively tune workflow parameters and workflow depth for typical phone usage patterns to avoid unnecessary computational load.

- *Enabling face-aware applications*: We propose two proof-of-concept applications built on top of the Visage system: 1) StreetView+, which makes use of the head pose inference from the Visage pipeline and provides user navigation on-the-go; and 2) MoodProfiler, which opportunistically senses users' expressions and provides visualized summaries while users are interacting with specific applications (e.g., email client and YouTube viewer) in their daily lives.

The rest of this paper is structured as follows. Section 2 describes the design considerations for the Visage system. Section 3 introduces an overview of the system architecture, followed by a detailed description of the Visage sensing, preprocessing, tracking and inference algorithms. Section 4 presents the implementation of the Visage system, while Section 5 summarizes the evaluation and benchmark results of the system. Section 6 applies the Visage engine to build two proposed face-aware applications. Section 7 reviews related work. Section 8 offers concluding remarks.

2 Design Considerations

2.1 User Mobility

Mobile users hold and interact with their phones in an uncontrolled manner and often in totally unexpected ways, such as, holding their phones at different angles or moving them around while doing various activities. These are new challenges not addressed by the computer vision community when designing face detection

and tracking algorithms [26] or computing the facial expressions of users [20]. Existing algorithms [2, 24] are designed and optimized for laboratory settings, where the camera is mounted and stable, with an angle that does not change over time. Because the front camera on the phone is subject to all degrees of freedom, we design Visage to make no assumptions about the camera motion, tilt and rotation relative to the user's face. The movement of the phone causes the image quality to drop in comparison to a mounted camera. Poor image quality will hurt all stages of the vision processing pipeline. Rather than adopting more sophisticated image processing techniques to address this problem (which would be inappropriate for resource-limited mobile phones), we propose taking advantage of accelerometer and gyroscope sensors to detect the tilt and motion of the phone. This design approach is computationally light in terms of resource consumption particularly in comparison to adopting more heavy duty signal processing techniques. As discussed in Section 5.3, Visage uses an opportunistic reinitialization scheme based on multimodal sensing for tracking that reduces the error to acceptable levels.

Furthermore, people take their phones wherever they go, ranging from bright outdoor areas through dark clubs and restaurants. Typically, computer vision based tracking algorithms assume consistent lighting conditions [26]. However, the lighting conditions in mobile environments change. To address this issue, Visage analyzes the exposure level of the local face region in the image rather than considering the entire image as default. Visage adaptively controls the front camera on the phone by maintaining a good exposure level in the face region of the image.

2.2 Limited Phone Resources

Video processing pipelines are computationally costly in general. The front-facing camera will produce 800 times and 50 times more data than those produced by the accelerometer and the microphone, respectively, in terms of the unit-time size of the raw data stream on an iPhone 4 mobile phone. One possible solution might be to off-load the interpretation pipeline to the cloud [29] and send inference results back to the phone. However, this leads to transmission energy consumption costs and variable delays in processing that complicates the development of continuous real time face-aware applications. In addition, users would have significant privacy concerns off-loading live videos containing their faces to the network or the cloud. To address the privacy issues and communication costs, Visage exploits a phone-based approach which processes video streams in a real-time manner (i.e., videos are neither off-loaded to the cloud nor stored on the phone). Although the processor speed and memory size found on the latest smartphones are increasing rapidly, they are still considerably limited in comparison to desktop or laptop computers. Importantly, phones are energy constrained (compared to a desktop), and video processing is the most computationally intensive task on the phone. Computer vision researchers focus on the development of algorithms [23] that deal with time varying high dimensional image data with little energy, processor and memory

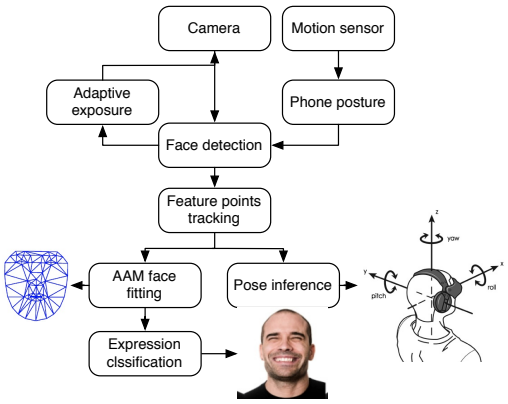


Fig. 1. Visage System Architecture

constraints. Visage is designed to operate in real-time while also addressing these constraints.

3 System Architecture Overview and Design

The Visage system architecture shown in Fig. 1 comprises sensing, preprocessing, tracking and inference stages. The *sensing stage* captures the video stream from the phone's front-facing camera and raw motion data from accelerometer and gyro sensors on the phone. The *preprocessing stage* comprises of three components: (1) phone posture, (2) face detection, (3) adaptive exposure components. By adaptively correcting the shaky/tilted frames (based on orientation and shakiness given by the phone posture component) and adjusting the image exposure setting (based on the illumination of face region given by the adaptive exposure component), the face detection component locates the user's face in the view. Given a face detected in the preprocessing stage, the *tracking stage* initializes a cylinder head model and continuously tracks individual feature points within the detected face region in all frames. If excessive motion is detected, Visage reinitializes the tracking module. By tracking relative locations of the feature points marked on the face, the user's 3D head pose can be inferred. Finally, in the *inference stage*, a shape and appearance fitting method is used to extract the face texture and structure features for facial expression classification.

3.1 Preprocessing Stage

Phone Posture Component. The preprocessing component identifies frames which contain the user's face, and monitors the phone posture (i.e., the tilt and motion status of the phone). Visage face detection and tracking are augmented with phone posture information to compensate the impact of the phone mobility.

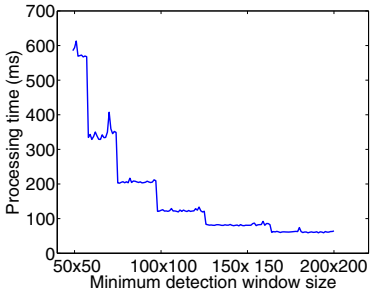


Fig. 2. Early termination scan scheme

This module takes raw readings of accelerometer and gyroscope sensors and estimates the direction of gravity and the intensity of motion from the variance of those two sensor readings, respectively. The system calculates the mean and variance on each direction at the same frequency as the video frame rate. The mean of accelerometer data provides a smoothed estimate of gravity direction. The variances of accelerometer and gyroscope data indicate the phone's motion intensity. They are used by the face detection and tracking modules to ensure system robustness.

Face Detection with Tilt Compensation. The process starts with locating the user's face in the first frame. This information is critical to the tracking task in consecutive frames. We use AdaBoost object detector [27] with tilt correction to make it robust to phone orientation changes, and to optimize the face-searching task by incorporating prior knowledge of the range of typical facial regions in images captured on mobile phones. The AdaBoost object detector operates by scanning the image using a large (e.g. 200×200 pixels) moving window. If no face is detected from a round of scans, the window size scales down by a constant factor of 1.2. The process terminates when a face is located, or scanning reaches the minimum window size. The algorithm normally starts from the largest possible detection window and scales down at each round by a constant factor to the smallest detection window, e.g. 20 pixels. In this paper, however, window sizes that are too small were omitted because the distance between the phone and the user's face is normally constrained by the user's arm length during mobile phone usage. This provides an estimate of the lower bound of the size of the user's face in the image, enabling us to prune scans with windows sizes that are too small. In our experiments, this resulted in an early termination scan scheme and higher system responsiveness. As illustrated in Fig. 2, on a 320×240 image, a 128×128 window size lower bound results in a detection time of about 80 ms, which gives more than 10 frames per second.

The standard AdaBoost object detector algorithm assumes that faces in the images are in an upright position, which is not necessarily true for phone cameras when a user holds the phone in a non-upright way. To work with possibly

tilted faces, two techniques are normally used: i) designing a set of classifiers handling each specific angle of tilted faces, or ii) using upright face detector on every possible angles. However, detectors trained for tilted faces are generally not as discriminative as the ones trained specifically for upright faces, and multiple detections incur more computational overhead. Thus neither technique is viable on mobile phones. Our solution leverages multi-modality sensing. With the input of accelerometer, Visage infers how the phone is tilted by estimating the gravity direction. The phone posture component calculates the gravity direction projected onto the coordinate system of the camera and feeds it to the face detection component. It is derived from smoothed accelerometer samples (i.e., a_x and a_y) of x and y axis on the phone by:

$$\theta_g = \frac{180}{\pi} \arctan \frac{a_x}{a_y} \tag{1}$$

Then the image is rotated by:

$$I_r = \begin{bmatrix} \cos\theta_g & -\sin\theta_g \\ \sin\theta_g & \cos\theta_g \end{bmatrix} I_i \tag{2}$$

where I_i and I_r are the matrices of original and corrected images, respectively. First, the tilt of the camera is compensated by the correction rotation. Then, the AdaBoost object detector is applied on the rotated image. The estimated gravity vector may include noise introduced by the user's movement. The AdaBoost object detector algorithm, however, has a tolerance window from about -15 to 15 degrees. It tolerates the rotation noise caused by the error of the estimated gravity direction.

Adaptive Exposure Component. In mobile environments, illumination is not always guaranteed and varies frequently. A clear face region is critical for tracking and inference. Proper exposures under dynamic illumination conditions play a key role. The default automatic exposure adjustment balances the exposure of the entire image, which results in underexposed faces in the captured images, as demonstrated in Fig. 3. When underexposed, certain details in the face region are lost, making the head pose tracking and facial expression classification more difficult. To address this issue, Visage uses the local lighting information within the detected face region to correct the camera hardware exposure level. After a face is detected, we calculate the local histogram H_{face} of the face window, and then determine its exposure level by computing the centroid of H_{face}:

$$C_{H_{face}} = \frac{\sum_{i=0}^{255} i H_{face}(i)}{\sum_{i=0}^{255} i} \tag{3}$$

where i is the intensity bins of the histogram. $C_{H_{face}}$ lying in the lower or higher ends of H_{face} would indicate the face being under- or over-exposed. This information is sent to the sensing module to adjust the camera exposure setting iteratively, until $C_{H_{face}}$ is in the center area of the histogram. After this step, the tracking and inference start.

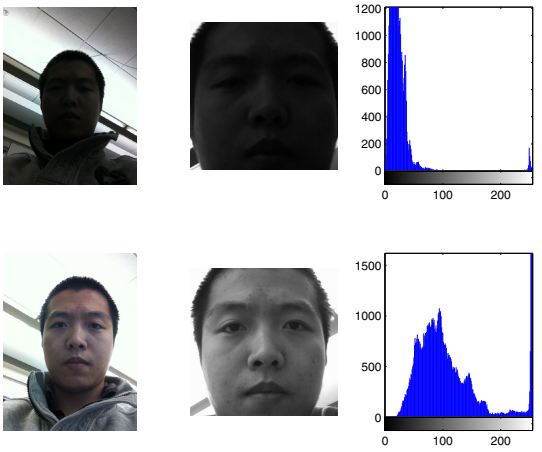


Fig. 3. Top: underexposed image, face region, and regional histogram; bottom: the image after adaptive exposure adjustment, face region, and regional histogram

3.2 Tracking Stage

Feature Points Tracking Component. After locating the bounding box of the user's face and setting the proper exposure level, this component initiates face-tracking algorithms using inter-frame information and estimates the 3D pose of the head. Feature Points Tracking Component first selects candidate feature points in the facial region from the first frame, and then tracks their locations in subsequent frames. Feature points represent landmarks on the facial object. Their spatial relationship to each other on the face is used to estimate the rotation of the user's head. Selecting proper feature points to track is the first step. Compared to points in textureless patches, points on object corners and edges (e.g., eye corners and edges of mouths) are normally selected as feature points because their salient visual features are stable across frames, and therefore, useful for tracking tasks. The level of textureness of a point is associated with the 2×2 autocorrelation matrix H of the second derivative image over a neighborhood W. If we let I_x and I_y be the gradients of the vertical and horizontal directions, respectively, then we have

$$H = \sum_{(x,y)\in W} \omega_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{4}$$

where $\omega_{x,y}$ is a Gaussian weighting term.

The smallest eigenvalues of H are sorted; the largest of these smallest eigenvalues correspond to feature points on corners and edges that can be tracked easily. We used the Lucas-Kanade method [3] (LK) to track the movement of the selected points in successive frames. LK assumes brightness constancy, that is, pixels in a tracking patch look the same over time.

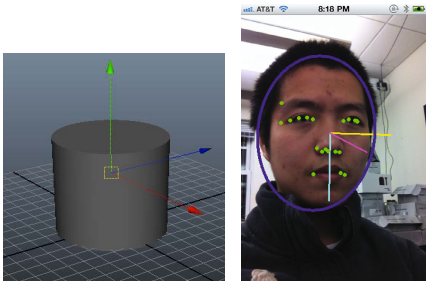


Fig. 4. (a) Cylinder model and (b) Tracking with pose estimation

$$I(x(t), t) = I(x(t + dt), t + dt) \tag{5}$$

where I is an image, t is time variable, and x is the pixel location. Assuming that a pixel's neighborhood has the same motion vector as the pixel itself, this vector (i.e., $[u\ v]^T$) can be expressed as:

$$\sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \sum_{(x,y)\in W} \begin{bmatrix} I_x I_t \\ I_y I_t \end{bmatrix} \tag{6}$$

where I_t is the derivative between images over time. Fitting the neighborhood to a least square estimator solves the motion vector ($[u\ v]^T$). In practice, a multilevel adaptive scheme is used to incorporate large motions between frames [3]. Feature points tracking provides precise details on the motion of the face, but the error of individual tracking points accumulates over time. This is a common problem of the LK method. Therefore, by using a more robust blob-tracking algorithm called CAMSHIFT [6], the tracking points are corrected if the phone motion is larger than the existing threshold. In CAMSHIFT, after the facial region in the bounding box is transformed from RGB space to HSV space, a histogram of the hue channel is calculated as a template. In the incoming video frames, a probability of face blob is calculated at every pixel by examining the counts in the histogram template at the location of the pixel's hue. The algorithm estimates the new location of the face by searching the region with maxima on the face probability image, and shifts the tracking blob to that location.

Pose Estimation Component. The Pose from Orthography and Scaling with ITerations (POSIT) algorithm [10] is used to estimate the 3D pose of the user's head. If an object's exact dimensions are known, POSIT is able to recover the six degrees of freedom movement of the object, with coordinates of at least four non-coplanar points on its 2D surface image and corresponding points describing its 3D structure.

To reconstruct the 3D pose, the algorithm requires the location of the 2D feature points on the image and their corresponding positions on the 3D head.

It is infeasible to pinpoint the 3D locations of the feature points on the user's face; thus, the human head is simplified to a rigid cylinder, as shown in Fig. 4. The diameter of the cylinder was set to the unit length. It is then straightforward to determine the 3D coordinates (X, Y, Z) of a 2D image point (x, y) on the initial frame. Because the initial frame contains a front face, coordinates in horizontal and vertical directions X and Y can be estimated directly by scaling image coordinates x and y, respectively; depth Z is obtained from the shape of the cylinder.

$$Z = \sqrt{r^2 - (X - r)^2} \tag{7}$$

After initialization, the 3D geometry of the points on the cylinder model is obtained. As the head moves in successive frames, the 2D feature points are tracked by LK. Using a rotation matrix and a translation vector to match the 3D cylinder points to the 2D image points, POSIT determines the directions for rotating the cylinder. Thus, the 3D pose of the cylinder is recovered, and the head pose is determined. POSIT requires at least four non-coplanar points; our study generated at least eight points.

The simplified cylinder face model introduces modeling errors because the surface of a face is different from that of a cylinder. Visage compensates for the modeling errors by a linear regression-based model calibration on all three rotation angles. On a given rotation angle, the calibration parameter is determined by

$$P_{calib} = (\Theta_{raw}^T \Theta_{raw})^{-1} \Theta_{gt} \tag{8}$$

where Θ_{raw} is a vector of raw output rotation angles, and Θ_{gt} is the corresponding ground truth angle. With calibration, the output angle in the inference step becomes:

$$\theta_{calib} = P_{calib} \begin{bmatrix} \theta_{raw} & 1 \end{bmatrix}^T \tag{9}$$

The inferred head pose can be used independently for subsequent components (or applications such as Streetview+) to detect when to trigger the eye region detection component.

3.3 Inference Stage

Active Appearance Model. To save computation, the following steps happen only when face orientation is determined as near frontal. The face features for classification in the system are representing both structure and texture of the user's face. They are obtained by Active Appearance Models [19] (AAM), a statistical method which incorporates texture and shape variations into the model to improve the accuracy. It describes the shape of a 2-dimensional image by a triangular mesh consisting of a set of landmark points. Active Appearance Models require a set of training images with a set of predefined shape-critical feature points. In particular, human facial expression recognition requires accurate feature points on human faces, such as edges and contours of eyes and the mouth. The shape of these feature points is the major component of facial expression. The texture model captures pixel color intensities to enhance the accuracy of model fitting.

By simultaneously considering the shape ($\mathbf{x}$) and texture ($\mathbf{g}$) of users' faces, an active appearance model can be synthetically determined with a combination of shape and texture models:

$$\begin{aligned} \mathbf{x} &= x_0 + Q_g c \\ \mathbf{g} &= g_0 + Q_s c \end{aligned} \tag{10}$$

where x_0 is the mean shape, g_0 is the mean texture, and Q_s and Q_g are variation matrices of shape and texture models, respectively. Instead of separating shape parameters from texture parameters, the combined model has a synthesized parameter (i.e., c) to describe the appearance of a human face. The training and fitting process is described as follows: (1) face images with predefined 20 landmarks are loaded from the face database to construct shape and texture vectors accordingly. (2) Principal Component Analysis (PCA) is then performed on normalized shape vectors and texture vectors.

Expression Classification. Visage uses both geometric and appearance features for facial expression classification as shown in Fig. 5. The combination of these two features makes Visage more robust to complex mobile environments. Visage recognizes seven expression classes: angry, disgust, fear, happy, neutral, sad, and surprise.

The face feature vector is constructed from the relative location of the 20 landmarks and the texture intensity within the tracking rectangle. We use a facial analysis technique called Fisher Linear Discriminant Analysis (Fisherface) [5] for the classification task. Fisherface seeks a projection direction (P) that maximizes the determinant of the between-class scatter matrix and minimizes that of the within-class scatter matrix:

$$P_{opt} = \arg\max_P \frac{|P^T S_B P|}{|P^T S_W P|} \tag{11}$$

where S_B is the between-class scatter matrix, and S_W is the within-class scatter matrix, respectively. S_B captures the variations between different facial expression classes, while S_W represents the variance within a given expression class, including different person identities, various lighting conditions, etc. The problems could be solved by a generalized eigenvalue solver.

$$S_B P = \lambda S_W P \tag{12}$$

where P is the projection matrix. All face feature vectors (i.e., I_{face}) are projected to this expression-specific subspace (i.e., v_{exp}) by

$$v_{exp} = P I_{face} \tag{13}$$

Given the predefined 7 classes of facial expressions, the size of the feature vector is reduced to 6 by the projection. The projected feature vectors are fed to a SVM classifier trained by LibSVM [8] in nu-SVC mode, with a RBF kernel and the nu parameter set to be 0.4.

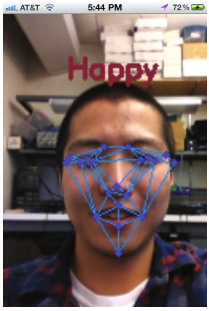


Fig. 5. Visage expression classification on the iPhone 4

4 Implementation

We prototyped the Visage architecture and algorithms on an Apple iPhone 4. Objective C is used to handle the GUI and hardware API. The core processing and inference routines are implemented in C. The OpenCV library [28] is ported to iOS as a static library, on top of which we built the Visage pipelines. The AAM fitting algorithm is ported and modified from VOSM [14] projects.

Standard picture resolutions (e.g., VGA (640×480) or higher resolutions) can be easily handled by desktop computers in real-time, but not mobile phones. Image size is an important factor impacting the overall computational cost, as demonstrated in Table 1. For phone-based systems, downsampling images to a lower resolution and skipping frames are inevitable. In theory, information loss caused by downsampling would degrade the performance. However, in practice, a phone's front-facing camera is usually positioned close to the user's face. Therefore, the size of the face regions captured by the phone's camera is much larger than those captured by distant cameras used in the surveillance or desktop cases. Therefore, downsampling introduces a much smaller performance penalty in our case. Studies show that face images smaller than 64×64 [9] lead to noticeable performance drops for expression classification tasks. To balance the computational load and the performance, Visage uses 192×144 as the operating resolution,

Table 1. Computational cost of face detection operations under different input image resolutions

Resolution	Time (ms)
640×480	4090
480×360	2123
320×240	868
192×144	298
160×120	203
96×72	68
80×60	53

Table 2. CPU and memory usage under various task benchmarks

Tasks	Avg. CPU usage	Avg. memory usage
GUI only	$< 1\%$	3.18MB
Pose estimation	58%	6.07MB
Expression inference	29%	4.57MB
Pose estimation & expression inference	68%	6.28MB

which normally contains faces of size around 64×64. At the same time, Visage also uses a frame skipping scheme where if the processing cannot keep up with the incoming frame rate, oldest unprocessed frames will be dropped. By doing this, the inference output is alway synchronized with the latest scene in the view range of the camera.

5 Evaluation

5.1 CPU and Memory Benchmarks

We evaluate the Visage system on the Apple iPhone 4 and present the detailed performance of each component of the multi-stage inference process.

Table 2 presents results of the CPU and memory usage under different face analysis tasks (i.e., benchmarks), while Table 3 shows the processing time of all Visage pipeline components. The face detection module is computationally expensive. But, after the first frame containing a face is identified by the detection module, Visage will track the detected face in consecutive frames without performing detection operations. When both detection and tracking modules are enabled, it takes approximately 53 ms to detect and track the face in a frame on average. The CPU usage for the iPhone 4 is about 68%, when the full pipeline is engaged.

Table 3. Processing time benchmarks

Component	Average processing time(ms)
Face detection	53
Feature points tracking	32
AAM fitting	92
Facial expression classification	3

5.2 Tilted Face Detection

To test the effectiveness of the accelerometer-augmented phone posture correction, we conduct the following experiment. Let the user hold the phone in his/her hand and rotate it during the face detection algorithms running on the phone. Images are captured when the phone is tilted by different degrees, as illustrated in

Fig. 6. The experimental results from two schemes (i.e., the default AdaBoost algorithm and proposed Visage's Face Detection with Tilt Compensation scheme) are reported with the rotation angles ranging from $-90° \sim 90°$, separated by an angle of 10° degrees. Detected faces are marked by red bounding boxes. The default algorithm fails to locate the user's face when the tilted angle is beyond the $[-15°, 15°]$ range. The proposed scheme is robust to the various degrees of tilted angles, far beyond the $[-15°, 15°]$ tolerance range of the default scheme.

5.3 Motion Based Reinitialization

The performance of feature points tracking is sensitive to camera movement. To test the benefit of our proposed motion based reinitialization, we conduct a field study and monitor the cumulative head tracking error of the feature points tracking with and without motion based reinitialization. In this experiment, we focus on the error caused by phone movements, therefore we keep the head still in an initial pose, in which case the ground truth is zero for all three rotation angles (i.e., yaw, roll and pitch). The estimated rotation angles of the head pose are recorded as well as the phone's motion intensity represented by normalized variance of accelerometer readings. Experimental results are shown in Fig. 7. Without motion-based reinitialization, after the phone stops moving, the head pose estimation drifts on all three rotation angles, and the estimation error accumulates over successive movements. With Visage's opportunistic reinitialization, the system automatically corrects the feature points tracking and cylinder model reconstruction errors whenever motion of the phone is detected. The tracking error is thus suppressed and does not accumulate over time.

5.4 Accuracy of Head Pose Estimation

In this first experiment, several evenly-spaced markers are placed along a circle of one-meter radius. Three volunteers participate in this experiment. Each of them is asked to stand at the origin of the circle and look at each of the markers. Ground-truth head rotation angles are measured by a protractor at the origin of the circle. Each participant contributes 5 samples for every marker. The head pose estimations without model calibration, calibrated readings with

Fig. 6. Images captured by the front-facing camera assuming varying phone tilted angles from -90 $\sim$ 90 degrees, separated by an angle of 15 degrees. The red boxes indicate the detection results. The first row is detected by the standard Adaboost face detector. The second row is detected by Visage's detector.

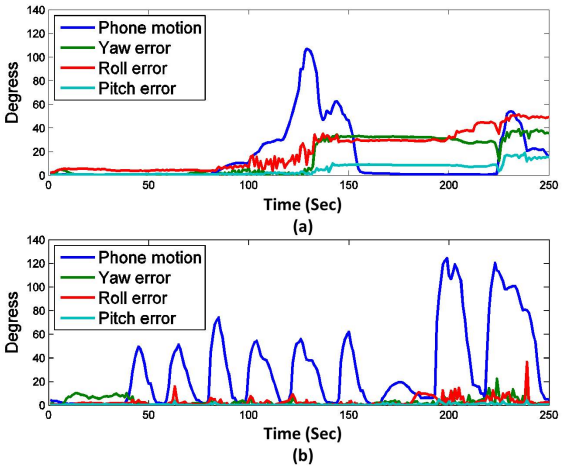


Fig. 7. Phone motion and head pose estimation errors (a) without motion-based reinitialization, and (b) with motion-based reinitialization

Table 4. Facial expression classification accuracy using the JAFFE dataset

Expressions	Anger	Disgust	Fear	Happy	Neutral	Sadness	Surprise
Accuracy(%)	82.16	79.68	83.57	90.30	89.93	73.24	87.52

model calibration, and the ground truth are shown in Fig. 8. With model calibration, the value of the mean absolute error drops by 60%, from $14.48° \pm 2.67°$ to $5.51° \pm 1.99°$.

5.5 Accuracy of Facial Expression Classification

To evaluate the accuracy of facial expression classification, we test Visage on two facial expression datasets (i.e., a public facial expression dataset and the other dataset collected by ourselves). In this experiment, five volunteers participate. Each one is asked to perform a list of predefined expressions. We capture 100 consecutive frames of each expression and use 5-fold cross validation. This experiment assumes that the facial expression on a mobile phone is personalized to its owner. So, for each user, the model is trained and tested by using the user's own training data. Table 4 shows the average expression recognition accuracy, with an overall accuracy of 83.78%.

To verify our method in more general cases, we test Visage's algorithm on the JAFFE [18] public expression dataset. JAFFE contains 10 subjects with 7 different expressions as in Table 5. Each subject contributes $3 \sim 4$ examples per expression. We carry out 10-fold cross validation on the entire dataset. The evaluation is repeated 10 times. Table 5 shows the confusion matrix of the classification results. The overall classification accuracy is 84.6%, which is comparable

to 81% reported in [25] and 85.6% in [15]. Note that our framework automatically detects faces, whereas [25, 15] used manually labeled face images.

6 Visage Applications

6.1 Streetview+

Streetview+ is a demo application built using the underlying Visage software engine. In this application, the user's head rotation with respect to the screen is tracked and used as an input source for users. Given the user's current location (i.e., longitude and latitude) obtained from the GPS sensor, the application shows the 360-degree panorama view with respect to that location from Google Streetview. The user's 3D head rotation is then used to change the viewing angle of the panorama as showed in Fig. 9. In practice, Visage achieves a tracking rate of around $12 \sim 15$ frames per second. Streetview+ provides an intuitive and realistic viewing experience in the virtual world, and the user can navigate the Google Streetview smoothly with continuous head movements.

6.2 Mood Profiler

The mood profiler application uses Visage to profile a user's mood in the background as they interact with different applications on their phone. When a user interacts with the phone, their realtime facial expression is monitored as well as the name and type of the foreground application. To reduce the resource usage, the emotion classification is performed only once per second. Fig. 10 shows the expression histograms of a single user using two different applications, the YouTube mobile app and the Email client. The histogram shows the subject's mood when using these two applications. In the case of watching YouTube videos, the subject manifests a wider range of expressions including happy, surprise, sad and disgust, while in the case of reading emails, the neutral expression is the dominant class.

Table 5. The confusion matrix of the facial expression classification based on the JAFFE dataset

Expressions	Anger	Disgust	Fear	Happy	Neutral	Sadness	Surprise
Anger	93.33	6.67	0	0	0	0	0
Disgust	6.90	75.86	17.24	0	0	0	0
Fear	0	7.41	92.54	0	0	0	0
Happy	0	0	0	87.10	6.45	3.23	3.23
Neutral	0	0	0	0	90.00	10.00	0
Sadness	0	6.45	9.68	3.23	9.68	70.97	0
Surprise	0	0	3.33	3.33	0	0	93.33

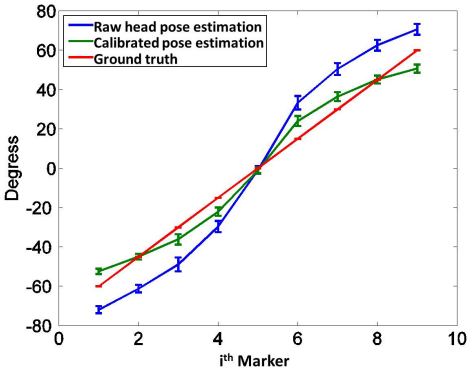
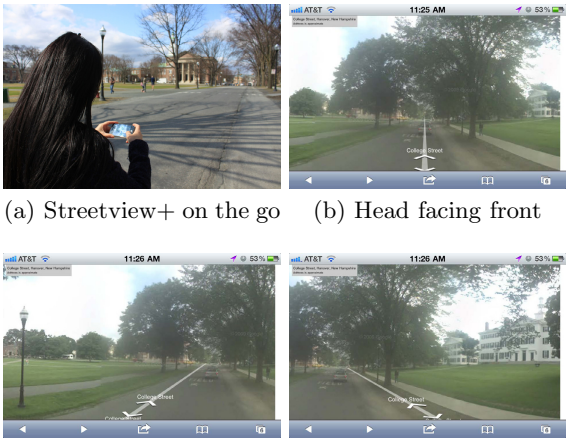


Fig. 8. Head pose estimation error



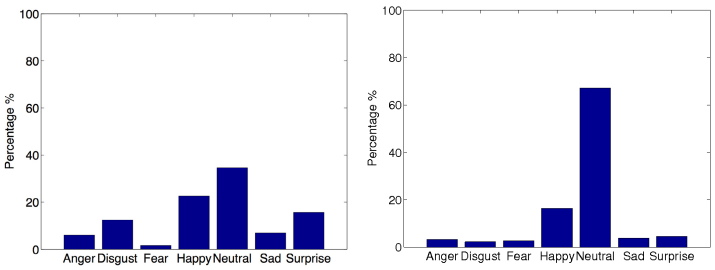
(a) Streetview+ on the go (b) Head facing front

(c) Head facing left (d) Head facing right

Fig. 9. Steetview+ enhanced with awareness of user head rotation

7 Related Work

There is a growing interest in applying computer vision algorithms in support of mobile applications [7]. SenseCam [12] is a life logging application. It takes pictures of the user's everyday life. However, it involves very limited image processing. MoVi [4] is a collaborative application where users collectively send images to backend servers to mine common interests. On the mobile side, Recognizr [1] is an application using the back camera to recognize objects. It connects the object's real identity with its virtual identity on the web. Recently, several content-based image retrieval mobile frameworks (e.g., Google Goggles) emerged. They usually require the user to take a photo of an object and then send the image or feature vector to a remote server for further processing.



(a) YouTube (b) Email

Fig. 10. Expression histogram when using (a) the YouTube mobile application and (b) an email client

The mobile phone is mainly used as a camera and the communication client with very limited local image processing. In contrast to these approaches Visage processes all the information locally on the phone - enabling, for example, head rotation as a novel UI for mobile phones.

In the HCI domain there is research that integrates the user's face into the mobile UI. PEYE [13] performs simple tracking of 2D face representations from the captured images; however, UI controls are limited to 2D directions (i.e., left, right, up, down) on the phone screen. Visage is capable of tracking the user's 3D head pose in a real-world coordinate system.

There is a considerable amount of research on head pose estimation [31] and facial expression analysis [31]. However, this work does not addresses challenges specific to mobile environments, e.g., camera motion, uncontrolled context, and computation efficiency. Much of the work achieved robustness by increasing the feature vector dimensions, finding more complex features. These purely vision-based approaches are typically computationally-demanding and not suitable for mobile phone applications. Littlewort *et al.* [16] proposed an expression classification algorithm for robots, however, the feature vector is based on pure texture instead of structure.

8 Conclusion

In this paper, we propose Visage, a face analysis engine specifically designed for resource limited mobile phones. Visage carries out all the sensing and classification tasks directly on the mobile phone without the need for backend server resources. In contrast to traditional mobile image analysis systems that rely on remote servers, Visage performs online processing at a lower computational cost but yields results that are comparable to existing off-line systems. Furthermore, multi-modality sensing is used to boost the system robustness in mobile environments. We present two proof-of-concept applications, and believe that the flexibility and robustness of Visage make it suitable for a wide range of face-aware applications.

References

[1] Recognizr, http://news.cnet.com/8301-137723-10458736-52.html
[2] Adiv, G.: Determining Three-dimensional Motion and Structure from Optical Flow Generated by Several Moving Objects. Trans. Pattern Anal. Mach. Intell. 7(4), 384–401 (1985)
[3] Baker, S., Matthews, I.: Lucas-kanade 20 Years On: A Unifying Framework. Int'l J. Comput. Vision 56(3), 221–255 (2004)
[4] Bao, X., Choudhury, R.R.: MoVi: Mobile Phone based Video Highlights via Collaborative Sensing. In: Proc. the 8th Int'l Conf. Mobile Systems, Applications, and Services, pp. 357–370. ACM, New York (2010)
[5] Belhumeur, P.N., Hespanha, J.P., Kriegman, D.J.: Eigenfaces vs. Fisherfaces: Recognition using Class Specific Linear Projection. Trans. Pattern Anal. Mach. Intell. 19(7), 711–720 (1997)
[6] Bradski, G.R.: Real Time Face and Object Tracking as a Component of a Perceptual User Interface. In: Proc. the 4th IEEE Workshop on Applications of Computer Vision, pp. 214–219. IEEE Computer Society, Washington, DC (1998)
[7] Chai, S.: Mobile Challenges for Embedded Computer Vision. In: Embedded Computer Vision, Advances in Pattern Recognition, pp. 219–235. Springer, London (2009)
[8] Chang, C.-C., Lin, C.-J.: LIBSVM: A Library for Support Vector Machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27 (2011)
[9] Cunningham, D.W., Nusseck, M., Wallraven, C., Bulthoff, H.H.: The Role of Image Size in the Recognition of Conversational Facial Expressions. Research Articles. Comput. Animat. Virtual Worlds 15(3-4), 305–310 (2004)
[10] Dementhon, D.F., Davis, L.S.: Model-based Object Pose in 25 Lines of Code. Int'l J. Comput. Vision 15(1-2), 123–141 (1995)
[11] Ekman, P., Friesen, W.V.: Constants Across Cultures in the Face and Emotion. Journal of Personality and Social Psychology 17(2), 124–129 (1971)
[12] Hodges, S., Williams, L., Berry, E., Izadi, S., Srinivasan, J., Butler, A., Smyth, G., Kapur, N., Wood, K.: SenseCam: A Retrospective Memory Aid. In: Dourish, P., Friday, A. (eds.) UbiComp 2006. LNCS, vol. 4206, pp. 177–193. Springer, Heidelberg (2006)
[13] Hua, G., Yang, T., Vasireddy, S.: PEYE: Toward a Visual Motion Based Perceptual Interface for Mobile Devices. In: Proc. of the 2007 IEEE Int'l Conf. Human-Computer Interaction, pp. 39–48. Springer, Berlin (2007)
[14] Jia, P.: Vision Open Statistical Models (2011), http://sourceforge.net/projects/vosm
[15] Liao, S., Fan, W., Chung, A., Yeung, D.-Y.: Facial Expression Recognition using Advanced Local Binary Patterns, Tsallis Entropies and Global Appearance Features. In: IEEE Int'l Conf. Image Processing, pp. 665–668 (2006)
[16] Littlewort, G., Bartlett, M., Fasel, I., Chenu, J., Kanda, T., Ishiguro, H., Movellan, J.: Towards Social Robots: Automatic Evaluation of Human-robot Interaction by Face Detection and Expression Classification. Advances in Neural Information Processing Systems 16, 1563–1570 (2004)
[17] Lu, H., Pan, W., Lane, N., Choudhury, T., Campbell, A.: SoundSense: Scalable Sound Sensing for People-centric Applications on Mobile Phones. In: Proc. the 7th Int'l Conf. Mobile Systems, Applications, and Services, pp. 165–178. ACM (2009)
[18] Lyons, M., Akamatsu, S., Kamachi, M., Gyoba, J.: Coding Facial Expressions with Gabor Wavelets. In: Proc. 3rd IEEE Int'l Conf. Automatic Face and Gesture Recognition, pp. 200–205. IEEE Computer Society, Washington, DC (1998)

[19] Matthews, I., Baker, S.: Active Appearance Models Revisited. Int'l J. Comput. Vision 60(2), 135–164 (2004)
[20] Michel, P., Kaliouby, R.E.: Real Time Facial Expression Recognition in Video using Support Vector Machines. In: Proc. the 5th Int'l Conf. Multimodal Interfaces, pp. 258–264. ACM, New York (2003)
[21] Miluzzo, E., Lane, N.D., Eisenman, S.B., Campbell, A.T.: CenceMe – Injecting Sensing Presence into Social Networking Applications. In: Kortuem, G., Finney, J., Lea, R., Sundramoorthy, V. (eds.) EuroSSC 2007. LNCS, vol. 4793, pp. 1–28. Springer, Heidelberg (2007)
[22] Mun, M., Reddy, S., Shilton, K., Yau, N., Burke, J., Estrin, D., Hansen, M., Howard, E., West, R., Boda, P.: Peir: The Personal Environmental Impact Report, as a Platform for Participatory Sensing Systems Research. In: Proc. the 7th Int'l Conf. Mobile Systems, Applications, and Services, pp. 55–68. ACM, New York (2009)
[23] Radovanovic, M., Nanopoulos, A., Ivanovic, M.: On The Existence of Obstinate Results in Vector Space Models. In: Proc. the 33rd Int'l Conf. Research and Development in Information Retrieval, pp. 186–193. ACM, New York (2010)
[24] Ristic, B., Arulampalam, S., Gordon, N.: Beyond The Kalman Filter: Particle Filters for Tracking Applications. Artech House Publishers (2004)
[25] Shan, C., Gong, S., McOwan, P.: Facial Expression Recognition based on Local Binary Patterns: A Comprehensive Study. Image and Vision Computing 27(6), 803–816 (2009)
[26] Szeliski, R.: Computer Vision: Algorithms and Applications. Microsoft Research (2010)
[27] Viola, P., Jones, M.J.: Robust Real-time Face Detection. Int'l J. Comput. Vision 57, 137–154 (2004)
[28] Willogarage, OpenCV (2010), `http://opencv.willowgarage.com/wiki`
[29] Yan, T., Kumar, V., Ganesan, D.: CrowdSearch: Exploiting Crowds for Accurate Real-time Image Search on Mobile Phones. In: Proc. the 8th Int'l Conf. Mobile Systems, Applications, and Services, pp. 77–90. ACM (2010)
[30] Yilmaz, A., Javed, O., Shah, M.: Object Tracking: A Survey. ACM Comput. Surv. 38 (2006)
[31] Zeng, Z., Pantic, M., Roisman, G., Huang, T.: A Survey of Affect Recognition Methods: Audio, Visual, and Spontaneous Expressions. IEEE Trans. Pattern Anal. Mach. Intell. 31(1), 39–58 (2008)

Multilevel and Secure Services in a Fleet of Mobile Phones: The Multilevel Secured Messaging Application (MuSMA)*,**

Serge Chaumette and Jonathan Ouoba

LaBRI, University of Bordeaux, France
{serge.chaumette,jonathan.ouoba}@labri.fr

Abstract. The level of guarantees that can provide systems leveraging MANets cannot reach that of more traditional networks due to their highly dynamic nature. Real MANets (based on mobile phones in our case) operate in an environment without any infrastructure where the nodes are intermittently connected over short or long periods of time. Solving the traditional problems that are identification, security of communications, content delivery and localization, therefore requires specific approaches. In addition, the mobile phones that make the MANet, offer different communication technologies that must be efficiently exploited in multilevel perspective, from the shorter range wireless radio standards to the longer range ones. This multilevel perspective must also integrate issues related to user preferences, energy saving and limited financial means. In this paper, we first propose a preliminary model taking into account all the elements previously mentioned in the context of a mobile phone oriented MANet. Next, we focus on two aspects that are the secure exchange of profiles and the choice of the transmission technology. We then describe the mobile application that we have developed based on this model and the first performance analysis that we have conducted. Finally, we detail the remaining challenges that must be solved in this novel approach and that will constitute the next steps of the research described in this position paper.

Keywords: MANet, multilevel communication, mobile phone, security.

1 Introduction

A MANet (Mobile Ad hoc Network), as considered in this paper, is composed of a set of communicating devices which are able to spontaneously interconnect without any pre-existing infrastructure and that configures itself on the fly. The devices (or nodes) can move around, appear, disappear what thus leads to modifications of the network topology.

* This paper presents a Work-in-Progress.

** This work is partly supported by the Smart Urban Spaces European project.

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 169–185, 2013.

It is then difficult to ensure certain properties in terms of security and authentication. We must decrease, compared to more conventional (infrastructure-oriented) networks, the level of guarantees we expect from the applications running on top of these networks. This is described in [1] where it is stated that it is impossible, for example, to maintain a single (connected) overlay over a MANet (what is a key feature in usual approaches to create services on top of mobile networks). Let us for instance consider authentication. Assume that within a MANet, two nodes meet, share their identities and communicate. Without the help of an external infrastructure, it is complex to authenticate the identities that were exchanged. However, it would be realistic to think of a system that would allow the two nodes, if they meet several times, to recognize each other. In some sense, to design systems adapted to MANets, we must mimic the behavior of people in real life. For instance, to get back to the previous example, people can talk to other people they meet in a crowd, even if they are not sure of their respective identities, and later recognize each other.

Existing mobile devices and mobile phones in particular, are endowed with several types of wireless technologies that expand and diversify their communication skills. The combined use of these technologies, while offering a variety of possibilities in terms of services and applications, requires a thorough analysis regarding security and the choice of the communication mode to use based on a number of criteria to be defined (for example energy cost, financial cost or privacy).

There are some noteworthy issues in this context of (mobile phones based) multilevel MANets. For example, in the context of the Smart Urban Spaces (SUS) European project [2] the goal of which is to propose new e-services, we intend to deploy what we call a Multilevel Museum Quest. The principle of the quest is to allow participants to answer, using a mobile application, series of questions disseminated inside a given museum. Some questions of the quest will concern pieces of art located in remote museums. For each question, either the player knows the answer and can go further in the quest, or he does not know the answer and then contacts others players (that may be in its immediate neighborhood or not) whose profile can suggest they might be able to answer the question he is trying to solve. So, the service must offer a player the possibility of: (i)discovering on the fly appropriate participants; (ii)determining the best way to contact them (Bluetooth, Wi-Fi, GSM, etc.). This application illustrates two of the basic problems to be solved within multilevel MANets. The question is thus how to allow a set of mobile phones to communicate securely using the most appropriate technology depending on the context? In order to contribute to solve this problem, our goal is to define a multilevel platform taking into account the different technologies available on the terminals that will, for the purpose of illustration, allow the secure exchange of messages. We study the theoretical and practical elements to consider in the design of the platform, model these elements, offer a reference application (secure exchange of messages) and validate the relevance of the proposed solutions.

The rest of this paper is organized as follows. First, we present our definition of the platform to implement and the key concepts to focus on. Then, we describe the first reference application that we have developed and the initial analysis that has been performed to evaluate it in terms of security, efficiency and energy consumption. We eventually present the remaining issues to address to make the platform effective, the extensions we propose for the platform and the future research work we plan to achieve, and we finally conclude.

2 Multilevel Framework

2.1 Overall Description

We first describe how we model the framework. The platform consists of a set of nodes equipped with certain types of wireless technologies (short, medium and long range) that can directly exchange messages (in a peer-to-peer manner with no routing procedures - one hop -). Each node has a profile (this will be discussed later) that contains its personal data, and we assume that the nodes have limited resources in terms of energy and storage capacity (these are mobile electronic devices). Our goal is to develop a flexible system that ensures the security of communications and the management of the multilevel aspect (choice of the most suitable technology to exchange messages depending on the context) still conveniently handling the personal data of the nodes.

Figure 1 shows a way of representing the global architecture and below we refer to it to describe the different entities and processes that are involved.

First, there are *the nodes and the communication technologies*. Each node supports a set of communication technologies and this set varies depending on the node. For example, in figure 1, n_1 is equipped with $Technologies$ 1 and 2 while n_3 is equipped with $Technologies$ 2 and 3. It is to be noted that the technologies have specific attributes, from the costs (energetic and financial) of transmitting a message, to their range and their transmission rate. In order to enable interactions between entities, the profile of each node specifies the characteristics through which it wishes (or does not wish) to be found, known or recognized by others. For example, in the Multilevel Museum Quest application that we have presented in the introduction, a participant may state he is an expert in a particular field of painting and thus express his willingness to help other players. Second, we have *the process related to the search for communication partners*. The search operation is based on the public information contained in the profiles. According to its needs (the case of the call for help in the Museum Quest for example) a node can specify with precise criteria the target node(s) it wishes to reach. Thus, for an efficient operation of the system, each node must publish as much as possible (in a coordinated manner in its different neighborhoods so that the duplications are minimized) the information it wishes to provide the others with. Hence, a node searching for a particular resource can explore the profiles that are accessible to it and try to find a node (or a set of nodes) that meets its expectations. In addition, when two nodes physically meet, the system must provide them with a way to exchange (if they wish) initial information in an easy

and secure manner (see section 2.2). This information is intended to allow the two nodes to recognize each other if they meet again (physically or virtually) or are private data (direct connection procedures for instance) in order to simplify any future communications between them.

Third, when a specified peer has been identified, *the communication between partners* can take place. In figure 1, the dotted lines, within each technology bus, represent the possible communication links that can be tied between nodes. We call a set of nodes capable of communicating with each other, at a given time, by means of the same technology (short or medium range) an *islet* of the system. One of the issue is to retrieve the technologies that are available on a node and that it can use to exchange messages. This is achieved using the information available in the profile of the considered node and by initiating *ping* procedures. *The technology to use for sending a message* to the node can then be chosen according to the constraints of the system (minimize the different costs) and the preferences of the involved entities.

It should be noted that we want to support not only direct communication but also another mean of communication, called *gateway mode* that allows a node to communicate with other nodes located in different *islets*. Concretely we call a *gateway node* (or gateway for short) a special node that authorizes the others to use it as a bridge to get access to remote *islets*. In figure 1, n_1 makes use of g_1 to reach the nodes of islet B. In order to use a gateway, a node must first subscribe to it.

The model of the framework highlights two essential elements within the system: the search for communication partners which is completed thanks to the information contained in the profiles; the absence of explicit localization of the nodes.

We can more precisely model the platform, which is graphically represented in figure 2, as follows:

- N is the set of nodes (mobile terminals) of the system. N contains the nodes n_i with $i = 1 \dots m$.
- T is the set of all available wireless technologies in the system. T contains the technologies t_j with $j = 1 \dots u$. The functions $rate$, $costE$, $costF$ are defined over T_{n_i}: $rate_{n_i}(t_j, t)$ = transmission rate for technology t_j at a given time t for the node n_i, $costE_{n_i}(t_j)$ = energy cost for sending a message via technology t_j for the node n_i and $costF_{n_i}(t_j)$ = financial cost for sending a message via technology t_j for the node n_i. It is particularly important to take into account the notion of time for the transmission rate due to its fluctuating nature. The function adr is also defined over T as follows $adr_{n_i}(t_j)$ = connection address to communicate with n_i through the technology t_j for the different entities of the system[1].
- $V_{n_i,tj}$, at a given time t, is the set of neighbors of n_i via technology t_j.
- $I_{t_j}(t)$, at a given time t, is an islet of the system. Each node n_i of I_{t_j} can directly send a message to another node of the islet by means of technology t_j. More formally, if $n_i \in N,\ n_i \in I_{t_j}(t) \Leftrightarrow V_{n_i,t_j} = I_{t_j}(t) - \{n_i\}$.

[1] We assume that this value does not change with time.

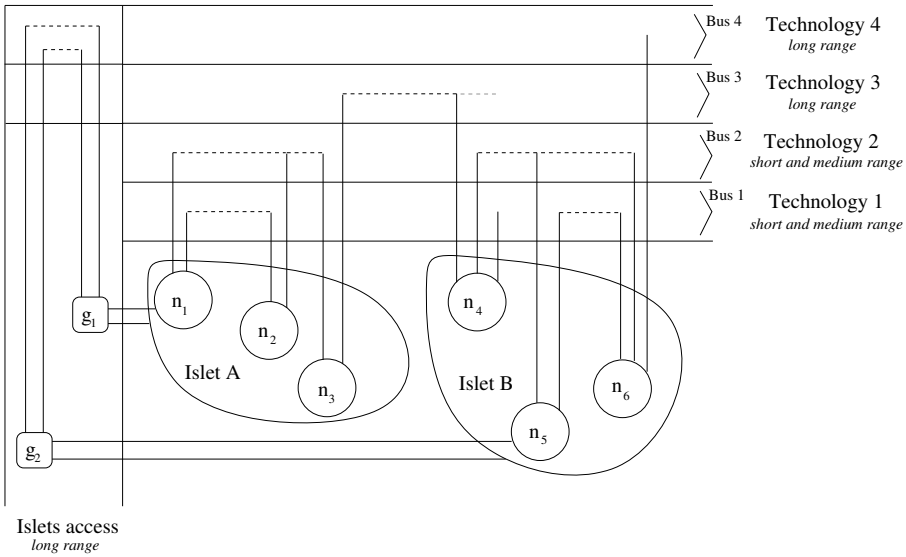


Fig. 1. Multilevel platform presentation

- G is the set of gateways, available in the system, allowing a node to communicate with remote islets (not directly accessible by using its embedded technologies). G contains gateways g_k with $k = 1 \dots b$.
- $n_i = (profile_{n_i}, techno_{n_i}, sub_{n_i}, pubKey_{n_i}, priKey_{n_i})$ with $profile_{n_i}$ the individual profile of the node, $techno_{n_i}$ the set of technologies it supports, sub_{n_i} the set of gateways (belonging to G) the node has subscribed to and $pubKey_{n_i}/priKey_{n_i}$ a RSA pair of public/private keys.
- we also define $kernel_{n_i}$ which are the minimal information to be exchanged (for node n_i) in case of physical meeting with another node.

  $kernel_{n_i} = (kernelprofile_{n_i}, adr_{n_i}(t_j), pubKey_{n_i})$ with $kernelprofile_{n_i}$ a subset of $profile_{n_i}$ and $adr_{n_i}(t_j)$ the connection information regarding the technology t_j (e.g. Bluetooth MAC address).

2.2 Concepts to Focus on

In this paper we will focus on two features of the platform. These are the elements that we were able, as of today, to test on a real application.

Secure Exchange of Minimal Information between Two Nodes. The goal of this process is to allow two nodes (n_i and n_j in the example), that physically meet and do not know each other, to securely exchange their kernel information (respectively $kernel_{n_i}$ and $kernel_{n_j}$). Figure 3 details the process that can be set up. The node (n_i) that initiates the action sends, through a dedicated channel, its public key and personal connection information (for a communication technology t_j it is equipped with, $adr_{n_i}(t_j)$ contained in $kernel_{n_i}$) to the

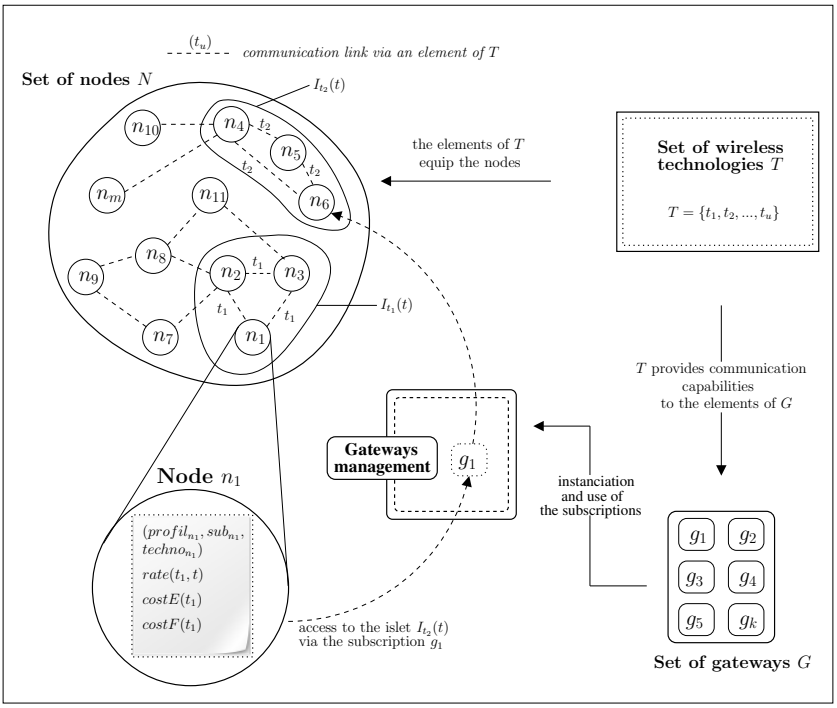


Fig. 2. Modeling of the platform

target node. The target node (n_j) transmits the information it wishes to share ($kernel_{n_j}$), ciphered with the public key included in the data that it just received, by using the personal connection information of the initiating node. In return, the initiating node sends the relevant information of its minimal profile ($kernelprofile_{n_i}$), ciphered with the public key of the recipient, through the same communication technology. The point to highlight is how the exchange process is initiated. The initial dedicated channel must enable a rapid and secure exchange of a limited amount of data while ensuring that the nodes are close to each other (in our prototype, it is achieved using Near Field Communication [3]). It is also important to note that due to the limitations of the dedicated channel (concerning the amount of data it can carry), it mainly serves to send the public key which will then help to secure the exchange of private information. Thus, apart from the dedicated channel, the protocol requires the use of another communication means. This is why the initiating node, in addition to its public key, sends its personal connection information for a communication technology, which communication technology must also be available on the second node. At the end of the exchange, n_i has received $kernel_{n_j}$ and n_j has received $kernel_{n_i}$. Following, the involved nodes will therefore be able to securely communicate with each other in order to share any kind of information, for instance information regarding their whole respective profiles.

It is worth mentioning that this exchange is not connected to the publication of profiles (that each node must perform) as described in section 2.1. Indeed, the publication of profiles was defined as a requirement for *the process related to the search for communication partners* to be effective. It is the dissemination of the profile of an entity to a set of nodes which can be contacted directly or through a gateway. In this case, the nodes do not physically meet and the published information is supposed to be public.

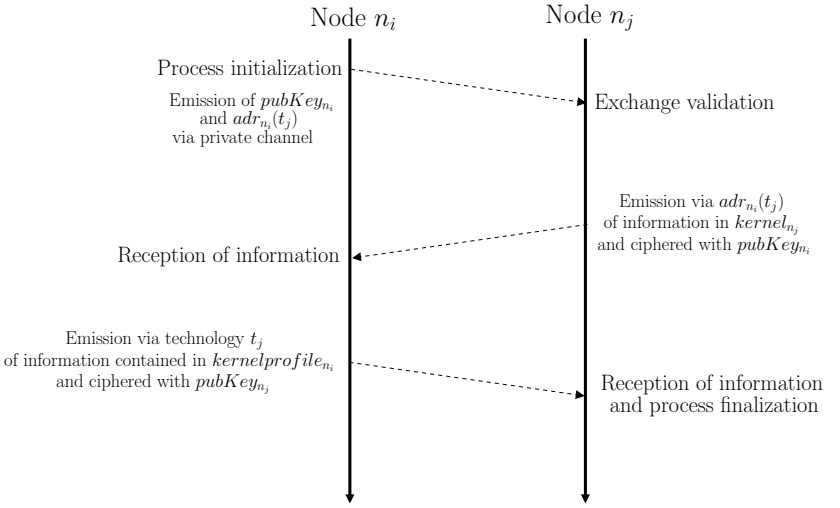


Fig. 3. Profile exchange process

Choice of the Transmission Technology. The problem to be solved at this level is: what technology t_j must be used by node n_i, at a given time t, to issue a message to another entity of the system so that the combination of $costE_{n_i}(t_j)$ and $costF_{n_i}(t_j)$ are minimal and $rate_{n_i}(t_j, t)$ maximal according to the different constraints? The constraints to take into account are those related to the profile of the considered sending node and the constraints of the entity to which the message must be sent. Indeed, a node can set in its profile some preferences in terms of energy and financial consumption or even have a limited number of technologies what impacts the way it sends or receives a message. Furthermore, the past activity of the system can also have an impact. It may be more appropriate to contact an entity of the system through technologies used in previous communications.

We thus need to compute the minimum of a cost function C (defined over T) with parameters representing the weighted factors to consider.

$C = \sum_{k=1}^{a}(w_k.cMin_k(t_i)) + \sum_{l=1}^{b}(\frac{w_l}{cMax_l(t_i)})$ with:

- $\sum_{k=1}^{a} w_k + \sum_{l=1}^{b} w_l = 1$ with w_k and w_l the weights of the factors
- $cMin_k(t_i)$ is a parameter to minimize (like $costE_{n_i}(t_j)$ and $costF_{n_i}(t_j)$) while $cMax_k(t_i)$ is a parameter to maximize (like $rate_{n_i}(t_j, t)$)

To compute $\min(C)$, we define three independent cost options: the *green option* where the node favors the situations with low energy consumption; the *cheap option* where the node minimizes as much as possible its financial costs; the *efficient option* when the transmission speed is essential. We could imagine, in the computation of $\min(C)$, that the weight 0.5 is assigned to $costE(t_j)$ if the node chooses the green option, to $costF(t_j)$ if the cheap option is selected and to $rate(t_j, t)$ in the case of the efficient option (for each option, the other parameters distribute the remaining weight (0.5) based on the different constraints).

Of course, a sending node must also be offered the possibility to arbitrary select the technology it wishes to use in order to send a message (thus bypassing the cost function based process) provided that the technologies in the corresponding entity are available.

2.3 Related Work

The multilevel platform model that we propose deals with many research issues that have often been explored but from different points of view. We have identified a few of them that are particularly relevant in the context of this paper.

Let us first consider the domain of network selection and cost functions where the goal is to be connected, depending on the context, to the most appropriate network in a seamless manner. In [4] and in [5] the authors present network selection strategies based on normalized utility functions that take into account parameters like bandwidth, power consumption and financial cost. Other network selection methods, with a policy specification model still based on cost functions, put more emphasis on specific issues such as the power-friendly aspect as in [6] or the user requirements as in [7]. Some authors also identify the handover decision (network selection) as a fuzzy MADM (Multiple Attribute Decision-Making) problem and propose a SAW (Simple Additive Weighting) based solution as in [8] and [9]. While taking into account the most essential elements, these solutions do not address the multilevel aspect of the platforms. Indeed, in our case, it comes to the special case of selecting the most appropriate intra-islet technology to enable peer-to-peer communication between two nodes, according to the constraints of the system. Our approach, by considering a multilevel architecture, is thus much more general.

Another noteworthy point is the concept of content-delivery in MANets. It is presented in various papers with a focus on a cooperative approach between nodes like in [10] or even with specific protocols for content-based communication as in [11]. These examples provide useful and relevant information for developing and deploying such networks. However, the multilevel aspect, that would consider the different technologies (from short/medium range to long range) available on a given node, is not handled. We claim that this is an important issue and we try to propose a solution for that.

Other approaches offer routing based solutions [12] [13] in the context of intermittently connected MANets based on collected mobility traces and estimations of the future behavior of the nodes of the network. Our approach with highly dynamic networks is to provide guarantees whatever the evolution of the network.

Simulation and traces thus do not help because they capture only one single execution path and cannot be used to prove results independently of the evolution of the network.

3 Reference Application: MuSMA

3.1 Architecture

The mobile application that we have implemented takes advantage of the availability of mobile devices (phones) equipped with at least four wireless technologies namely NFC, Bluetooth, Wi-Fi and GSM.

NFC [14], which stands for Near Field Communication, is a communication technology that has a range of about 10 centimeters. Indeed, because of its very short range, which requires entities wishing to communicate to be physically close to each other (which provides security guarantees), it offers an adapted way of starting the kernel information sharing process (see section 2.2).

Bluetooth, Wi-Fi and GSM are the elements of T among which a choice is performed to transmit a message. It is to be noted that when we mention the GSM technology it comes to sending messages via SMS (Short Message Service) and for Wi-Fi it is the direct mode [15], which enables direct peer-to-peer communications, that we consider.

Figure 4 shows the components of the application. The user interaction takes place through the GUI (Graphical User Interface) that lets the user enter the messages to send, view the received messages as well as the different profiles. The Personal Profile stores the private data of the user and its contacts. A key point lies in protecting the application with a password (selected by the user upon first use) which limits the access to sensitive information like the private key which is itself ciphered thanks to the above password. Then, there is the Technology Selection module which is responsible for selecting the most appropriate technology according to the cost evaluation strategy (section 2.2). It is also responsible for notifying the user of the technologies that are available to communicate with a given contact. It is accomplished by using the connection information (Bluetooth, Wi-Fi, GSM) contained in the profile of each contact; it helps to periodically perform ping requests, for each type of technology and for each contact, to collect the related connectivity status. The Exchange Protocol implements the protocol defined in subsection 2.2 so that the user can easily add a new contact. Finally, the Communication Management module interacts with the different available wireless technology layers to send and receive data.

3.2 Prototype Implementation

Our system requires the use of mobile phones that support NFC and more precisely its peer-to-peer mode (for direct communication between the phones) and the Wi-Fi direct standard. The phones must also be able to send messages via Bluetooth and GSM (SMS). The Android platform which is one of the most

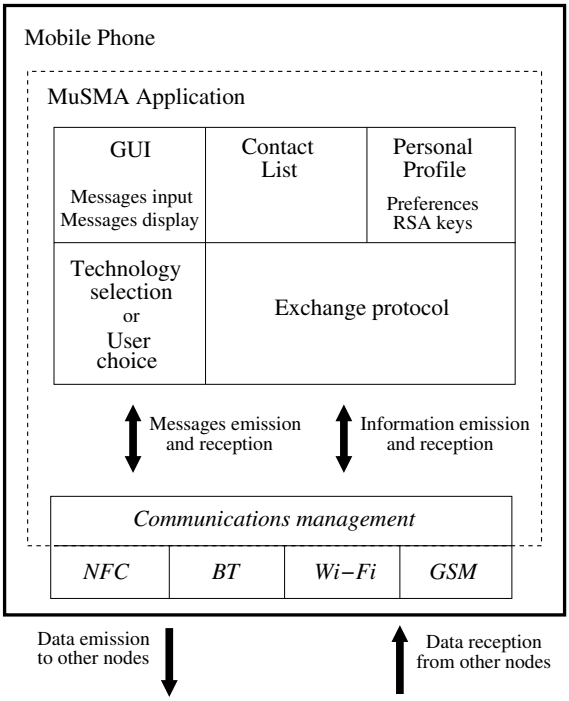


Fig. 4. MuSMA application architecture

dynamic at the moment, offers, in cooperation with the manufacturers, some phones that correspond to the required specifications. Therefore, we decided to target this platform which also has the APIs (in its 4.0.4 version) and provides a (limited and uni-directional) support for NFC peer-to-peer and Wi-Fi direct modes. Then, a first prototype has been developed to analyze and validate some aspects of the architecture proposed for MuSMA. The mobile used for this development is the Galaxy Nexus. A simple usage scenario of the application is described figure 5. In addition, figure 6 presents a screenshot of the current prototype where a user is willing to send a message to one of its contacts.

4 First Analysis of the Solution

The goal is to provide a first evaluation of the mobile application that we have developed. This first analysis must show the applicability of the multilevel framework (at least some aspects) to a real world mobile application in terms of efficiency and energy consumption while the application should ensure that the security requirements are guaranteed. It should be noted that usability is a key element in any mobile application. However, it is not considered in our analysis due to the fact that the user interface is based on the Android built-in SMS

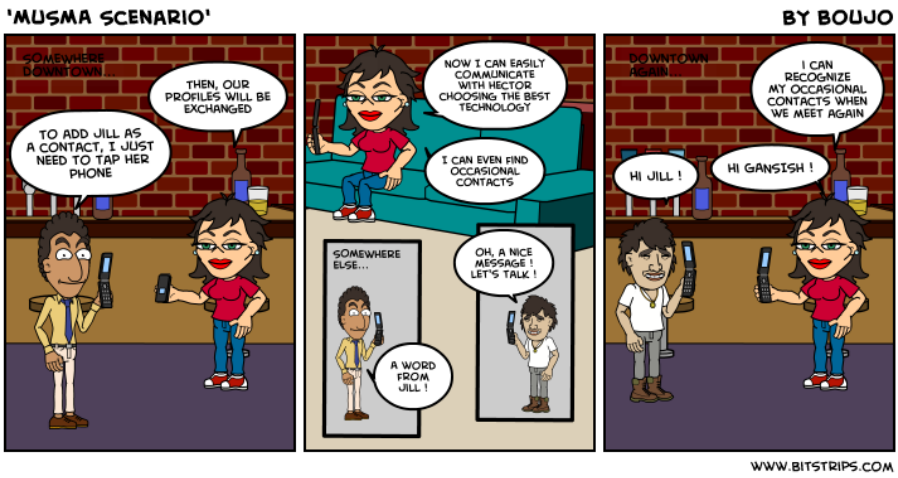


Fig. 5. Scenario for the tests

messaging application. Then, in our opinion it is not necessary to focus on this concept, at least for now.

4.1 Security

The security of the system, once the kernel information has been exchanged, relies on public/private keys based protocols where the confidentiality of the exchanges between the nodes are ensured (the authentication and integrity aspects are not considered for now). Of course, these procedures are dependent upon (i): the secrecy of the private key, but they also rely on (ii): the properties of the communication technologies that we use. Thus, we will focus on these two points.

As each instance of the application (with its user) acts as a node of the system, it is provided with a pair of RSA keys. The public key is provided to the external contacts while the private key is used to decipher incoming messages issued by known entities. The secrecy of the private key is entrusted by the fact that the access to the mobile application is protected by a password. Upon first use, the legitimate user chooses a password to proceed to a symmetric ciphering (with the AES protocol in our case) of the private key that is then stored in the mobile phone. Then, when launching the application, the user is prompted to enter a password which is used to decipher the private key. A random number is generated, ciphered with the public key (which is also stored in the phone) and deciphered with the private key resulting from the entered password. This results in another number. If the two numbers match, the access to all the functionalities of the application is granted. Hence, the secrecy of the private key is built on top of the strength of the underlying symmetric algorithm. However, when the application is running the private key is loaded to the memory of the phone to decipher incoming messages. It can cause security holes because the mobile phone is not considered a trusted device

Fig. 6. Screenshot of the prototype application

(malicious programs can corrupt it). We must consider a stronger solution where the private key would be stored in a secure element [16] ((U)SIM card, microSD smart card, etc.) that will perform the ciphering and deciphering operations. We have not tested the improved solution yet.

Concerning the kernel information exchange presented in 2.2, it is essentially the NFC technology which is concerned. Since our approach makes use of the NFC peer-to-peer mode, we must consider its threat model. There are attack scenarios such as eavesdropping, man-in-the-middle, and relay [14] [3]. All of them are not to really significant in our context. For example, the man-in-the-middle attack is difficult to set up in a real NFC-based environment [3] as well as the relay threat [17]. This is due to the nature (very low range) of the NFC technology that requires both transacting parties to be physically located next to each other. Thus, the main concern lies in the eavesdropping threat. An attacker could eavesdrop [18] and record the data of the transactions, namely the public key and the personal data of a node. The consequences of this attack are limited because either very few personal data or data intended to be public are transmitted through this channel. However, this issue could be solved by using the SNEP protocol [19] (which is unfortunately not yet implemented on the phones we have) to establish bi-directional NFC peer-to-peer connections in the initialization of the exchanges.

Even if the system can be enhanced, it nevertheless reaches a reasonable security threshold for our messaging application.

4.2 Energy

We wish to verify that the additional energy consumption induced by the use of the application is not too high. We have performed some tests to evaluate this consumption overhead. To do so, we have defined a specific usage profile of the application that is based on realistic statistics. Indeed, based on the average number of SMS sent [20] per month in Europe in 2011 (whereby in some countries users can issue up to 20 SMS per day), we assumed that a user could send 60 messages (20 via Bluetooth, 20 via Wi-Fi and 20 via GSM) per day using our application by taking into account the three available technologies. In a concrete way, we fully charged the battery of some phones that we used in a conventional manner recording the energy consumption. We repeated the same procedure under the same conditions and with the same phones but in addition running our mobile application (according to the previously defined usage profile). Figure 7 presents the results that we obtained with 3 phones. The overhead seems

	% Battery used	% Battery used with MuSMA	Overhead
Mobile 1	24	28	16.6%
Mobile 2	26	31	19.2%
Mobile 3	34	40	17.6%

Fig. 7. Energy consumption overhead for issuing the 60 messages

significant and it can be explained by the fact that the Bluetooth, the Wi-Fi direct and the GSM technologies must be permanently activated. Nevertheless, it must be put in perspective relatively to the consumption of other applications, for example the android OS built-in browser drains 0.35% of the battery for a 30s run [21].

5 Other Issues Regarding the Platform

Obviously, the next step in our work will be to go deeper in the modeling of the platform we are building. To do this, it will be necessary to detail in a systematic way the issues regarding the publication of profiles within the system as well as the discovery of the nodes and their recognition. It will also be fundamental to ensure a better selection of the technology to communicate with based on more criteria (the past activity for instance). This must be done bearing in mind that the proposed solutions must be tailored to fit the context of mobile phone oriented MANets. Indeed, in the context of MANets, the most critical issue is not to ensure that two nodes can always find a route to communicate with each other. The dynamic and mobile nature of the nodes makes it difficult

to implement such protocols in realistic applications. It is rather to ensure that, depending on the context, the possibilities of communication offered to each node of the system are optimal.

Concerning the publication of profiles, it is related to the process which manages *the search for communication partners*. As we already mentioned, each node of the system must publish as much as possible the information (about its profile) it wishes to provide the other with. The traditional approach which consists in flooding the system (each node that receives the information transmits them to all its neighbors) should be avoided because it is assumed that the nodes have limited resources in terms of energy and storage capacity (and broadcast storms should also be avoided). Each node must then send the profile to its neighborhood while paying attention not to send it several times to the same recipient (at least on the part of the network where the radio broadcast is not the underlying implementation). This must be done by keeping track of the nodes which received the considered profile. Each node must also make use of relay nodes available in its neighborhood and that are equipped with longer range technologies to reach inaccessible areas for the transmission of its profile.

The specification of the targets is also related to *the search for communication partners*. Indeed, this process allows to define a set of nodes that are targeted for a particular operation. This specification is based on a description of characteristics that must be common to all the targeted nodes. Concretely, the selection of a set of targets is carried out by means of an exploration request whose main argument is the characteristics sought among all the accessible nodes. The specification may involve one or more nodes. For example, if an entity wishes to try to get in touch with a node whose pseudonym it knows, it must initiate a exploration request with the pseudonym as an argument. This request causes the system to search, among the profiles that the requesting node can access, those containing the relevant characteristic. Depending on the process used for publishing profiles, we could for instance state that the accessible nodes to an entity that initiates an exploration request are the ones it received, the ones of its neighbors or those which are reachable through a relay node. This has to be explored further.

On the topic of improving the method for selecting the transmission technology, it mainly concerns *the technology to use for sending a message*. The selection method is based on the minimum of a cost function as presented in 2.2. It is therefore a question, for a given node n_i, of defining the appropriate strategy for computing the weights and the values for the parameters (such as $costE_{n_i}(t_j)$ and $costF_{n_i}(t_j)$) of the cost function while considering the factors that influence the constraints and their sets of definition. The factors to consider include the following: the technological preferences, the priorities in terms of cost and transmission rate, the profile of the sending node and the profile of the recipient.

We summarize figure 8 the operations within the system that are concerned with the presented perspectives. In the example, the node n_1 sends its profile and an exploration request to its neighbors. The neighbors are divided into three categories which are not disjoint: the group of neighbors accessible via the

communication technology t_1, the group of neighbors accessible via the communication technology t_2 and the group of neighbors accessible via the communication technology t_3. The relay nodes n_9 and n_{10} that are equipped with longer range communication technologies are used to transfer the messages to the entities to which n_1 has no direct access. This scenario puts forward the elements to consider in order to enhance the modeling of the platform. We will then incorporate all the improvements of our model in the mobile application that we develop. In order to get more valuable results in the analysis of our system, we plan to test our application on field and on a larger scale which may be considered through the deployment of the museum quest application (presented in the introduction) in some European cities (members of the SUS project). Another notable point is the fine grained energy profiling of our mobile application that must be performed to enhance its efficiency.

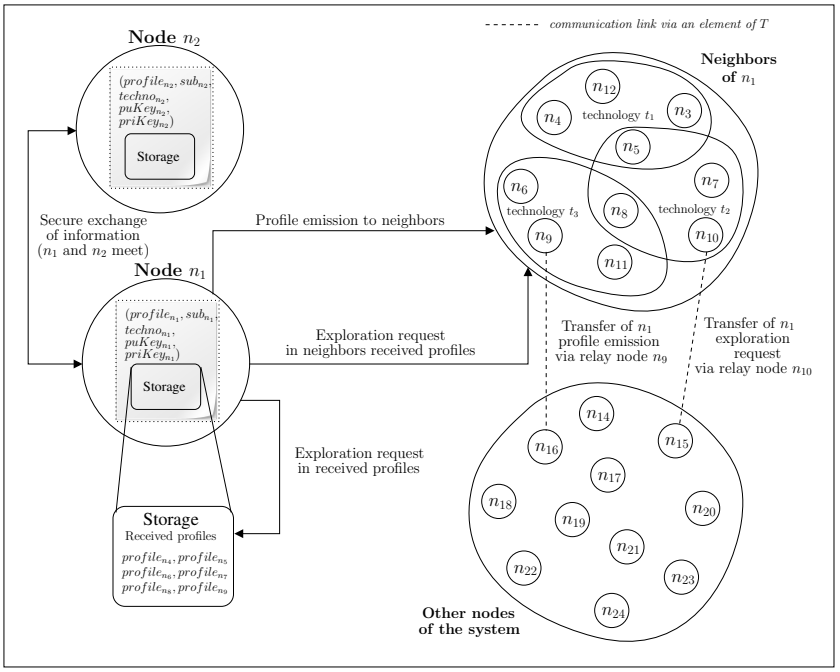


Fig. 8. Overview of operations within the platform

6 Conclusion

In this paper, we have presented our initial work on a model to support multilevel secure communications within mobile phones oriented MANets. The multilevel aspect, leveraging the technological capabilities of mobile phones, leads to the exchange of messages in peer-to-peer manner with the selection of proper communication means according to the profile and the availability of the considered

entities. This model, with its focus put on the search for communication partners via the profiles and the non-explicit localization of nodes, attempts to provide a more realistic approach of the highly dynamic nature of MANets. We have also presented and briefly analyzed (in terms of security and energy) the first mobile application that we have developed based on the model. The MuSMA application has made it possible to validate the feasibility of certain elements of the model including the secure exchange of minimal information and the choice of the transmission technology.

These initial results encourage us to pursue the definition, the improvement and the extension of our model while considering to use it with concrete cases such as the Multilevel Museum Quest application in the framework of the SUS European project. In addition, the description that we proposed concerning the remaining issues to solve (namely the publication of profiles, the specification of the targets and the improved method for selecting the transmission technology) draws clear directions for future work.

Acknowledgment. The work presented is this paper is carried out within the framework of Smart Urban Spaces, an ITEA2 [22] European project, the goal of which is to define new e-services for cities. The proposed services mainly take advantage of specific technologies, in particular NFC, in order to ease the everyday life of European citizens. Several use cases are concerned, for instance daycare organization, transportation or cultural events attendance. We would like to thank all the partners of the project with whom we have been working.

References

1. Chaumette, S.: Can highly dynamic mobile ad hoc networks and distributed MEMS share algorithmic foundations? In: Proceedings of the 2nd Workshop on Design, Control and Software Implementation for Distributed MEMS, Besancon, France (2012)
2. Smart urban spaces website (September 2010), `http://www.smarturbanspaces.org`
3. Haselsteiner, E., Breitfuss, K.: Security in Near Field Communications (NFC). In: Proceedings of the Workshop on RFID Security (2006)
4. Wang, H.J.: Policy-enabled handoffs across heterogeneous wireless networks. In: Mobile Computing Systems and Applications, WMCSA, pp. 51–60 (1999)
5. Serrador, A., Correia, L.M.: Policies for a cost function for heterogeneous networks performance evaluation. In: Proceedings of the 18th Annual IEEE International Symp. on Personal, Indoor and Mobile Radio Commun., Athens, Greece (2007)
6. Trestian, R., Ormond, O., Muntean, G.-M.: Power-friendly access network selection strategy for heterogeneous wireless multimedia networks. In: 2010 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting BMSB, pp. 1–5 (2010)
7. Shen, W., Zeng, Q.-A.: Cost-function-based network selection strategy in integrated wireless and mobile networks. In: Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops, Washington, DC, vol. 02, pp. 314–319 (2007)

8. Zhang, W.: Handover decision using fuzzy MADM in heterogeneous networks. In: 2004 IEEE Wireless Communications and Networking Conference IEEE Cat No04TH8733, pp. 653–658 (2004)
9. Bari, F., Leung, V.C.M.: Automated network selection in a heterogeneous wireless network environment. IEEE Network, 34–40 (2007)
10. Ma, Y., Jamalipour, A.: A cooperative cache-based content delivery framework for intermittently connected mobile ad hoc networks. Trans. Wireless. Comm., 366–373 (2010)
11. Haillot, J., Guidec, F.: A protocol for content-based communication in disconnected mobile ad hoc networks. Mob. Inf. Syst., 123–154 (2010)
12. Lindgren, A., Doria, A., Schelén, O.: Probabilistic routing in intermittently connected networks. SIGMOBILE Mob. Comput. Commun. Rev., 19–20 (2003)
13. Spyropoulos, T., Psounis, K., Raghavendra, C.S.: Efficient routing in intermittently connected mobile networks: the multiple-copy case. IEEE/ACM Trans. Netw., 77–90 (2008)
14. Madlmayr, G., Langer, J., Kantner, C., Scharinger, J.: NFC devices: Security and privacy. In: Proceedings of ARES (2008)
15. Alliance, W.: Wi-Fi certified Wi-Fi direct: Personal, portable Wi-Fi technology. WIFI Alliance. Tech. Rep. (October 2010)
16. Reveilhac, M., Pasquet, M.: Promising secure element alternatives for NFC technology. In: 1st International Workshop on NFC, Hagenberg, Austria (2009)
17. Francis, L., Hancke, G., Mayes, K., Markantonakis, K.: Practical NFC Peer-to-Peer Relay Attack Using Mobile Phones. In: Ors Yalcin, S.B. (ed.) RFIDSec 2010. LNCS, vol. 6370, pp. 35–49. Springer, Heidelberg (2010)
18. Hancke, G.P.: Practical eavesdropping and skimming attacks on high-frequency RFID tokens. Journal of Computer Security 19(2), 259–288 (2011)
19. N. Forum, Simple NDEF exchange protocol - SNEP 1.0, NFC Forum, Tech. Rep. (2011)
20. GSMA, A. Kearney, and W. Intelligence, "European mobile industry observatory", GSM Association, Tech. Rep. (2011)
21. Pathak, A., Hu, Y.C., Zhang, M.: Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof. In: Proceedings of the 7th ACM European Conference on Computer Systems, pp. 29–42. ACM, New York (2012)
22. Itea2 website (September 2010), `http://www.itea2.org/`

vUPS: Virtually Unifying Personal Storage for Fast and Pervasive Data Accesses

Mohammed A. Hassan, Kshitiz Bhattarai, and Songqing Chen

Department of Computer Science,
George Mason University
{mhassanb,kbhattar,sqchen}@gmu.edu

Abstract. More and more overlapping functions on all kinds of mobile devices with these on traditional computers have significantly expanded the usage of mobile devices in our daily life. This also causes the demand surge of pervasively and quickly accessing files across different personal devices owned by a user. Most existing solutions, such as DropBox and SkyDrive, rely on some centralized infrastructure (e.g., cloud storage) to synchronize files across different devices. Therefore, these solutions come with potential risks of user privacy and data secrecy. In addition, continuously maintaining strong consistency among multiple replicas of a file is very costly.

On the other hand, today a common user often owns sufficiently large storage space across her personal home desktop, office computer, and mobile devices. Therefore, in this paper, we aim to design and implement a system to *v*irtually *U*nify *P*ersonal *S*torage (vUPS) for fast and pervasive accesses of personal data across different devices. vUPS provides similar services as offered by existing cloud-based storage services, but (1) vUPS consists of only personal computers without involving any third party, thus it minimizes the risks of user privacy and data secrecy; (2) vUPS organizes all storage in a distributed fashion so that it is not prone to the single point of failure; (3) vUPS differentiates files and maintains different consistency policies in order to reduce the consistency maintenance cost. Having implemented vUPS with HTML5, we conduct extensive experiments to evaluate its performance. The results show that vUPS offers similar user performance when compared to DropBox.

1 Introduction

With the ever-increasing processing power and ever-decreasing prices, mobile devices are getting more and more popular. According to International Data Corporation, the total number of smartphones sold in 2010 was 305 millions [11], which is a 76% increase from 2009, and there are already over 4.6 billion mobile subscribers in the world and the number is still growing [12]. The prediction is that there will be around 982 million smartphones in 2015 [11].

To some extent, today mobile devices are replacing their counterparts for Internet accesses, such as emailing, web surfing, and Internet entertainment.

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 186–204, 2013.

For example, mobile video traffic is now over 50% of the mobile data traffic on the Internet, and it is predicted to occupy 2/3 by 2015 according to Cisco [3].

Such a trend has also caused the demand surge of pervasive data accesses, such as for photos, across a user's different storage space, such as her iPhone and desktop computer. To respond to such a fast increasing demand, the current research and practice have provided many solutions, such as `Dropbox` [5], `Google Docs` [8], `Amazon s3` [2], `Windows SkyDrive` [14] and `SME Storage` [15]. They mainly rely on cloud-based services or a server-based approach.

These centralized cloud or server-based approaches [2] [5] [14] typically require a user to store all files on the storage owned by the service providers, which risks data security, privacy, and availability. For example, it has been reported on the news about Mark Zuckerberg's pictures leak incident in Facebook [13], DropBox account breach with wrong passwords [6], Amazon's data center failure in 2011 [1], etc. Some modern file systems [15] [32] [41] have taken into account user's storage to avoid third party storage compromise. But they maintain a strong consistency model for different types of files, resulting in unnecessary and heavy performance overhead. For example, smartphones are often associated with consuming and producing data which are mostly non-editable (photo, music, and video) and more than 27% pictures are taken by smartphones [4]. For these files, a strong consistency model is an overkill.

On the other hand, today an average user typically possesses multiple computers, such as personal home computers, office computers, and mobile devices. While the storage of one device at a time may not be sufficient for storing all data files of the user, the total storage space is often large enough to store all files owned by the user. Even when the total storage space of a user is not large enough, it is relatively cheap to buy additional storage nowadays as one-time cost.

Therefore, in this study, we aim to design and implement a system to *v*irtually *U*nify *P*ersonal *S*torage (vUPS) for fast and pervasive file accesses. With vUPS, all participating devices of the user are transparently and seamlessly unified. A file can be accessed through a web browser, considering that today the web browser is the vehicle for end-user's accesses. In vUPS, there is no central node that maintains the file system states. Instead, any device can serve as the server when it is actively used. To minimize data transferring, only meta-data is proactively accessed while data files are accessed in an on-demand fashion. In vUPS, different types of files are treated with different consistency policies in order to balance the consistency and the maintenance overhead.

To evaluate vUPS, we have implemented a prototype based on HTML5 and Javascript. The prototype provides a standard API for other web and desktop applications to access and manipulate vUPS data. We have conducted experiments with micro-benchmarks and also compared to DropBox. The results show that vUPS can offer a similar user experience to DropBox.

In summary, we have made the following contributions in this paper.

- vUPS provides an alternative solution to existing cloud-based or other centralized approaches for responding to the demand surge of quick and pervasive file accesses across multiple devices owned by a user.
- By differentiating and treating different types of files, vUPS strives to achieve a balance between the file consistency and the maintenance overhead.
- With a web browser interface and a standard file access interface, vUPS can be adopted by other applications to transparently and seamlessly access personal files.

The rest of the paper is organized as follows. We discuss the design of vUPS in section 2 and the consistency models in section 3. vUPS implementation is presented in section 4, and we evaluate its performance in section 5. Some related work is discussed in section 6 and we make concluding remarks in section 7.

2 vUPS Design

In this section, we first briefly overview the architecture of vUPS. Then we discuss its components.

2.1 vUPS Overview

Instead of a centralized architecture, vUPS adapts a flexible P2P architecture. Figure 1 illustrates the architecture of vUPS that runs across multiple devices, such as a user's home computer, an office computer, a laptop, and a smartphone.

In vUPS, the participating devices peer with each other to virtually form **a** single and global storage space. Communications (e.g., to fetch data or to

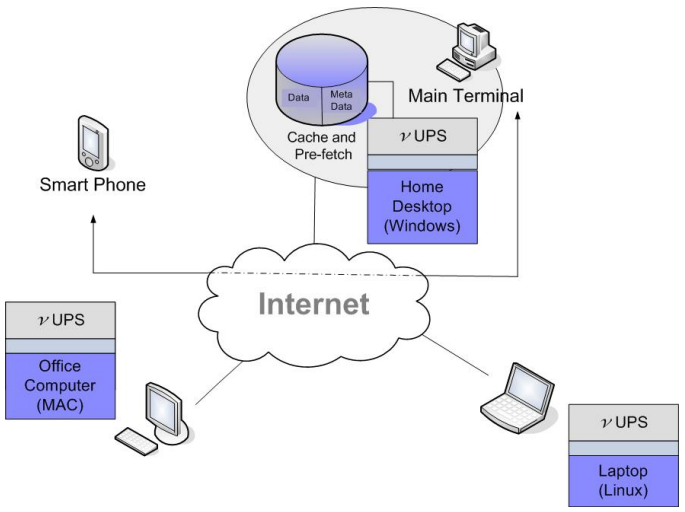


Fig. 1. The Architecture of vUPS

execute a user's commands) between devices are based on RESTful web services [25]. With such a P2P architecture, the device that the user is actively using as the main device (i.e., on which the user initializes her request) is responsible for supporting the main functions of vUPS. For this reason, we refer to this device as the `main terminal` of vUPS. Currently, at each time, there is only one main terminal. As the principal component of the vUPS, the main terminal is responsible for maintaining the vUPS namespace. In our design, the vUPS namespace is stored in a logically separate site. Once the main terminal is activated upon user accesses, the namespace will be loaded. In practice, this namespace can always be stored on the user's home desktop computer. All other participating devices are referred to as passive terminals, as they are mainly responsible for responding to the requests from the main terminal. In addition, users and other applications can interact with vUPS via vUPS APIs. Currently, we have designed a web-based user interface based on vUPS APIs.

Note that when a user actively uses her mobile device as the main terminal, it can deplete the limited battery power soon, because we expect that the main terminal could be relatively stable and stay on-line for a long time. Thus, in vUPS, the main terminal functions can be delegated by the mobile device to a more powerful computer, such as the home desktop (as shown in Figure 1) or the office computer.

When an application or a user needs to access a file, vUPS first finds the real device hosting the file based on the proper mapping in the vUPS namespace. vUPS resolves this mapping via the user name, device ID, resource path and operation type as: http://<usr>.vUPS.com/DeviceID/Path&Operation. Note that in the case of a delegated main terminal, the actual file transferring happens directly between the two involved devices without involving the main terminal.

2.2 vUPS Components

To support the desired functionalities, we design vUPS with the vUPS API, the vUPS Controller, the vUPS Synchronizer, the vUPS Data Manager. Figure 2 depicts the design with these major components. Next we discuss each of these components.

- *vUPS API:*
  To provide an interface of vUPS for users and applications, we develop vUPS API. These vUPS APIs support typical file operations, such as create, delete, read, write, as well as typical directory operations. To provide an easy access to users, we further develop a Web browser based user interface based on vUPS APIs. It uses HTML5 and Javascript to make it accessible from heterogeneous platforms such as smartphones, desktop and office computers that may run different operating systems.
- *vUPS Controller:*
  The vUPS controller consists of two modules. The first one is the bootstrapping module or bootstrapper. Basically, when the main terminal is accessed by the user, it first needs to load the vUPS namespace, which contains the

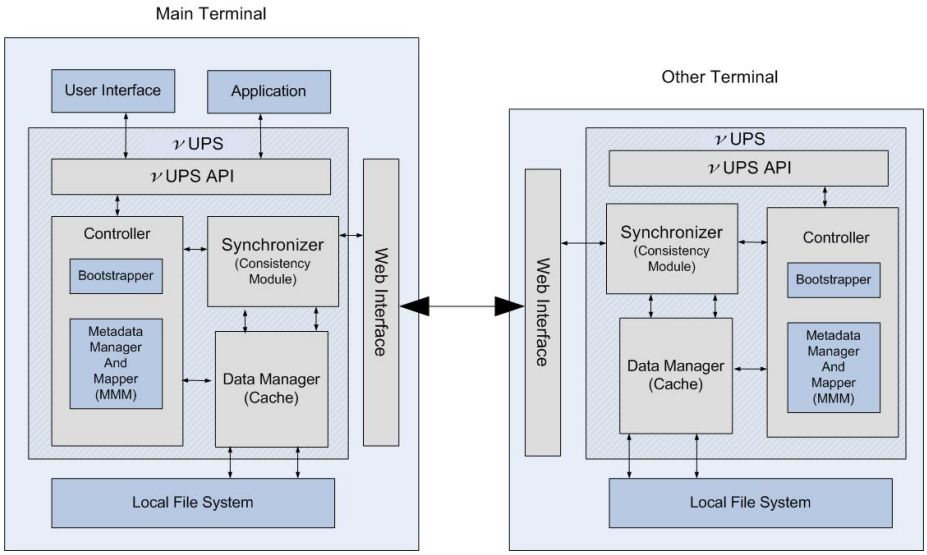


Fig. 2. vUPS Components

metadata and the directory hierarchy of the vUPS file system. When a new device wishes to join the system, it also provides the bootstrapping information so that any device can be added or removed from the system.

The second module is the metadata manager and mapper (MMM). Metadata is crucial to vUPS and strong consistency is maintained for metadata. MMM also maps files in the vUPS system to their physical locations in different devices. When a file is created by the user, a new entry is created to the appropriated location in the namespace. Accordingly, a file deletion removes the entry from the namespace.

– *vUPS Synchronizer:*
Maintaining consistency is essential to any distributed file system. vUPS Synchronizer is responsible for consistency maintenance among multiple copies of a file on different devices. As we shall discuss later, vUPS uses a non-traditional consistency model based on the file types in order to balance the overhead and the performance. Note that because the namespace is loaded on each participating device, the vUPS is also responsible for maintaining the consistency of namespace.
– *vUPS Data Manager:*
The data manager deals with the physical storage of the data in the local disk via traditional filesystem APIs. A particular notable function of the vUPS Data Manager is to manage cache for better user response time. As the core of the cache management, a LRU (Least Recently Used) replacement policy is used for cache management.

These components work together in a collaborative manner. Basically, the namespace of the vUPS is often stored in a reliable device or a web site if a bootstrapping site is used. A participating device needs to contact this bootstrapping storage to load the initial namespace as well as the main terminal address.

When a file operation takes place (either by the user or by some application) through vUPS API, vUPS API passes the request to the vUPS Controller, which finds the appropriate file through the MMM.

The MMM then checks with the Synchronizer for cache validation. For read operation, the Synchronizer checks the cache and validates the cache if it is found in the cache. If it is not valid, then the Synchronizer finds the physical location of the data from the MMM. Then the Synchronizer either contacts the local or remote Data Manager to fetch the data. A remote Data Manager is invoked via the web interface. Whenever a request is received from the web interface, the Synchronizer enforces the operation via the local Data Manager. If the operation is write, then the Synchronizer finds the devices that contain the replicas and/or the cached copy of the data. Then the Synchronizer updates its local cache and data (if any) through the Data Manager. It also propagates the update to all the relevant devices via appropriate web service interfaces.

Note that vUPS relies on a storage place to store the initial namespace. This single point of failure can be eliminated by replicating the namespace among all the devices. This may however increase the maintenance overhead.

3 Consistency and Availability

Achieving availability and strong consistency at the same time is a major challenge in a distributed file system [30]. Traditional file systems replicate the data across multiple devices to increase availability, which also increases the overhead for consistency. Popular file systems have different policies to make the data consistent among replicas. For example, Coda [39], Ficus [27], and Ivy [33] apply optimistic concurrency control between the replicas. Coda resolves consistency conflict using version controlling. Bayou [43] provides high availability with a weak consistency model. Bayou was mainly designed for intermittently connected devices, which is also suitable for mobile devices. With pervasive network connections today, it is ideal to use continuous synchronization and have quick consistency. For example, cloud storage like Google Drive [9] uses strong consistency, while DropBox [5] applies continuous consistency between the replicas.

Both the optimistic and the continuous consistency models suffer from the scalability issue. In addition, continuous consistency also suffers from the overhead for continuous updates. That is, even a byte change in the data may result in a lot of network traffic (several thousand more than the update itself). Such network overhead is amplified by the number of devices in the system.

Observing the fact that strong consistency is not required for every type of files, vUPS has different replication and consistency policies for different types of files. Today a user often owns much more media data (e.g., audio, video, and

image) files than before. For example, it has been shown that a typical Windows user has 11.6% image files of the total storage [21]. This number is increasing due to the vast use of smartphones and tablets, because they are generally used to take pictures or videos. On the other hand, the documents and other files only consist of a small portion of the storage (i.e. 2.5% of the total file system [21]). Note that media files are not frequently changed. Only the corresponding metadata (favorite, ratings, etc.) may be modified. Therefore, we may relax the consistency model for media files as they are often non-editable, while the replicas of other documents have to be consistent as they are frequently edited. In addition to that, fetching the document files on demand is not expensive compared to that for media files as the media files size [16] is often bigger than that of documents files [21] on average.

Thus, in vUPS, files are divided into two categories: editable and non-editable. The audio, video, image, and program files are considered as non-editable. On the other hand, all other files, including doc, xml, and text files, etc., are categorized as editable files. Note that these non-editable files can also be updated very occasionally, but vUPS only maintains weak consistency among their replicas. In the current implementation, vUPS differentiates different types of files based on their filename extensions.

With two categories of files, vUPS has the following two different policies for different types of files: (1) limited availability with strong consistency; (2) high availability with weak consistency.

3.1 High Availability with Weak Consistency

Considering the different types of popular files [22], we categorize the popular video, audio, and image files as non-editable files. Although in our current design these files are recognized solely based on file extensions, any self-learning clustering algorithm can classify the files over time based on the modification history. These non-editable files are seldom modified. Thus, the access to update rate is often high for these files. Moreover, the size of these media files is often larger than the editable files such as doc or xml [16] [21]. Caching and replicating these files on every possible device improves the access time to those files and results in less network traffic, but it also increases the overhead of maintaining consistency between copies.

As these files are seldom modified, vUPS follows a weak consistency model between the copies. A user can request the file from any device. The device may contain the data locally or fetch it from other devices (through the main terminal) and cache it. Whenever a modification takes place in a device, the change is reflected on the local/cached copy. The user does not need to wait for the update being propagated to other devices. Similar to Bayou, vUPS propagates updates during pairwise contact. This epidemic approach ensures that as long as the participating devices are not permanently disconnected, the writes will eventually reach every copy [20].

For non-editable files, vUPS follows an invalidation protocol to notify the changes to other copies. As an invalidation protocol only notifies the changes, it

saves network traffic and latency as the real update is not sent. The application in the devices may or may not pull the update for that file, depending on the policy of that application. vUPS aims to support application-specific conflict resolution, and each application may provide specific dependency check for updates. If the dependency check is valid, then the update takes place. Otherwise, conflict arises and the updates are either merged or marked invalid. Unlike scheduling or database transactions, media files may not have conflicting updates. Thus the merging procedure for media files ensures the latest update to a particular file is applied when any conflict arises. To detect the latest update, vUPS has vClock (vUPS vector logical clock) for each file and each folder. Each file has a vector of the Lamport logical clock [31], where each entry in the vector represents the timestamp for each device associated with that file. Whenever a file is created, a vector timestamp is created with entries for each device where the file is replicated. In addition to that, whenever that file is cached, an additional timestamp entry is added to that vector. If a cached copy is deleted or a replica is removed, the corresponding entry is removed from the vector. For every update from any device, the logical clock for that device in the vector is incremented. Whenever a device joins vUPS, all the data are updated to the latest vector. Thus, the copies are always up-to-date and synchronized if at least one copy of the data is always online. If only one copy of a file is kept connected to the vUPS and all the other copies are offline (that is, only one copy is mutual-exclusively online), then the file is updated if the timestamps of all the vectors are greater than the previous values. If all the entries in the vector are smaller than the new values, then it is not updated. Otherwise, the file is marked as conflicted and both copies are kept for the user to resolve manually. Thus, vUPS ensures the total ordering for the updates between the devices provided at least one copy is always online.

Note that, for non-editable files, the metadata may be changed frequently (a user may change the rating of movies, tag of pictures, etc.), but vUPS considers metadata as editable data, for which strong consistency is maintained among replicas.

Whenever a non-editable file is invoked by a device, the file is cached on the device and in the main terminal according to the caching policy. As the access-to-update ratio is higher, leaving a copy behind will improve the access performance [42]. So, when that file is closed, it is synchronized (if necessary), and a copy is left in the cache. The namespace contains a callback reference to that address for providing response from that copy to other devices.

3.2 Limited Availability with Strong Consistency

For copies of the editable files, vUPS maintains strong consistency. As strong consistency is not scalable, vUPS maintains a minimal number of copies to maintain the availability and consistency to get the best of both worlds.

Similar to non-editable files, whenever an editable files is accessed by a device, the file is cached on the device and in the main terminal according to the caching policy. All the modifications take place on the cached copy and are propagated to remote copies (local read/remote write). When that file is closed, it is synchronized

(if necessary), and the local cached copy is deleted. The callback reference is also deleted from the namespace once the file is closed. That is, vUPS enforces a strong consistency policy between the copies. It sends the update operation where data should change (active replication).

As the access-to-update ratio is lower, keeping a local copy close to the device may not be better [42] as it may send frequent updates to all the replicas while these updates may not be accessed by the user. In this case, keeping sending these updates can waste bandwidth and may not be scalable. So it deletes all the cached copies once the read/write operation in the cached copy is completed. This approach makes vUPS more scalable. As the average size of editable files is smaller than that of non-editable files, bringing these files to the cache when needed does not affect the performance too much. In addition, for these types of files, a typical replication policy results in less network traffic.

Assume the failure probability of one machine is p and the number of replication is r. Then, the failure probability of all replicas is:

$$M(n,p,r) = (p^r \times \sum_{i=0}^{n-r} p^i \times (1-p)^{(n-r)-i}) \tag{1}$$

Let the unavilability probability of one machine due to network is q and the number of replication is r. Then, the unavailability probability of all replicas due to network failure is:

$$N(n,q,r) = (q^r \times \sum_{i=0}^{n-r} q^i \times (1-q)^{(n-r)-i}) \tag{2}$$

$M(n,p,r)$ and $N(n,q,r)$ are independent from each other. So, the availability of the over all system is:

$$A(n,p,q,r) = 1 - (M(n,p,r) + N(n,q,r) - M(n,p,r) \times N(n,q,r)) \tag{3}$$

The typical disk failure rate is 4% [7]. If we take this into account, the availability of vUPS can be deduced to be more than 96% for a typical system with four machines with no replication. We can get more than 99% availability by keeping a backup (a replica) of each file, which is comparable to amazon services (with a failing rate 0.1-0.5% [7]).

In case of a terminal failure, the active requests for that terminal is re-directed to other terminals with replicated copies. When the device joins vUPS again, the replica in that device is checked again (with SHA-1) with other replicas and synchronized if necessary.

vUPS is also highly robust in the sense that any terminal can be initiated, allocated, deleted or backed up without interrupting the system. vUPS also allows adding or deleting a disk, allocating storage, moving data, or interrupting system availability. This design allows all data to be remained online and available to clients and other terminals.

4 vUPS Implementation

We have implemented a prototype of vUPS with Java and HTML. The user may add devices with any operating system and can browse, access, and store data in the vUPS storage consisting of those devices. In addition to the user interface, we have implemented the vUPS APIs in Java for applications. Figure 3 shows a screen-shot of vUPS user interface.

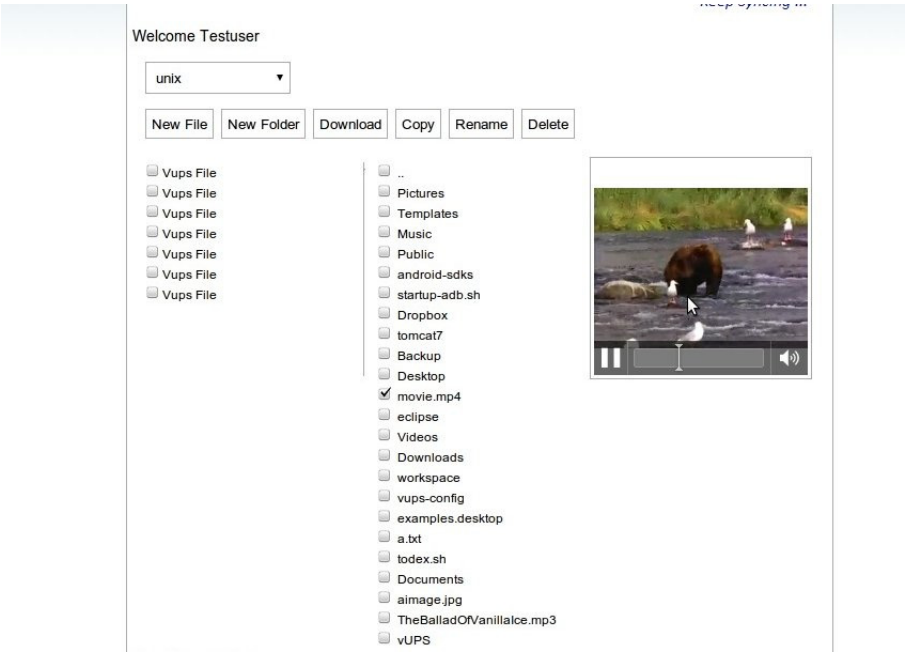


Fig. 3. vUPS User Interface

The *vUPS API*s are built with Javascript and HTML5. A local web server is used for bootstrapper right now where a user registers her devices. After login, the user is provided with the main HTML page where they may select any devices, access data, copy between devices, etc. These operations are implemented by Javascript to call the web services in the devices. The *Metadata Manager and Mapper (MMM)* manages the mapping between the vUPS files to their physical locations that are saved in a SQLite database.

The *vUPS Synchronizer* communicates with the devices through the RESTful architecture as mentioned earlier. The *Data Manager* gives an abstraction over the local file system with the underlying physical storage in the local disk via Java filesystem APIs.

For mobile devices, we also build a vUPS app to access the data. There is also vUPS API for Android to access vUPS from other apps. The current prototype

maintains minimal security over the RESTful architecture in HTTP. We use the IP address of the device as the device ID in the current prototype implementation.

In the current implementations, files are simply differentiated based on their name extensions. Extensions like .mpeg, .wmv, .dat, .mov are considered as the popular movie filename extensions. vUPS includes the .mp3 and .mid as the popular audio files, where the .jpeg, .jpg, .gif, .bmp, and .png are considered as filename extensions of image files. Files with these name extensions are all considered as non-editable files. By default, all other files are considered to go under frequent modifications, and vUPS have different policies for them.

5 Performance Evaluation

In this section, we evaluate the performance of vUPS based on the prototype we have implemented. In the experiments, we first use the micro-benchmarks to evaluate vUPS. Then we compare the performance of vUPS with the popular application Dropbox.

In these experiments, we configure vUPS with three commodity desktops, one laptop, and one smartphone. To compare with the service provided by DropBox, we have also set up a DropBox account.

All the desktop machines have 8 GB memory and 2.7 GHz CPU. The laptop has 4 GB memory and 2.7 GHz CPU. The desktop machines run Windows 7, Ubuntu 11 and Mac operating systems, respectively. We use a Google Nexus One phone with 512 MB memory and 1 GHz CPU running Android 2.3 operating system. The DropBox account has 2 GB of storage.

5.1 File I/O Performance

First, we study the performance of file reading. In vUPS, a desktop is designated as the main terminal. Files of 1 KB, 10 KB, 50 KB, 100 KB, 200 KB, 500 KB, 1 MB, and 3 MB are read by the smartphone via the main terminal. To emulate the realistic scenario, files are randomly distributed on these devices. When a read request for a file is sent from the smartphone to the vUPS, the main terminal receives the request from the smartphone. The main terminal searches the namespace for the ID of the device that stores the file, and forwards the web address of the resource to the smartphone. The smartphone then completes the read operation by bypassing the main terminal. The results are compared against when a direct/local read from the sdcard of the smartphone and when the file of the same size is located on DropBox. We take the average of 100 runs for each file size and the results are shown in Figure 4. The x-axis represents the file size. The y-axis shows the response time. We set the network speed as 500 Kbps for local accesses. The maximum and the minimum response time are within 2% of the average with 95% confidence level. As expected, the read time increases with the increase of the file size and neither vUPS nor DropBox is comparable to the local read performance. However, vUPS constantly outperforms DropBox, for

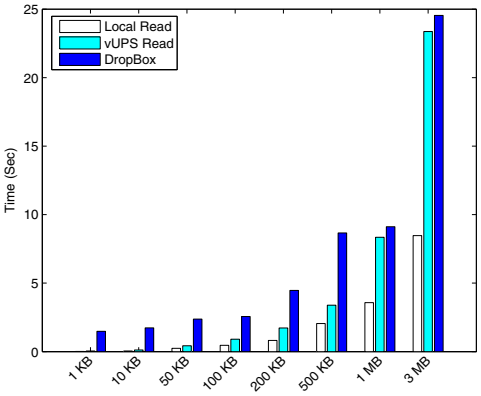


Fig. 4. File Size vs. Response Time

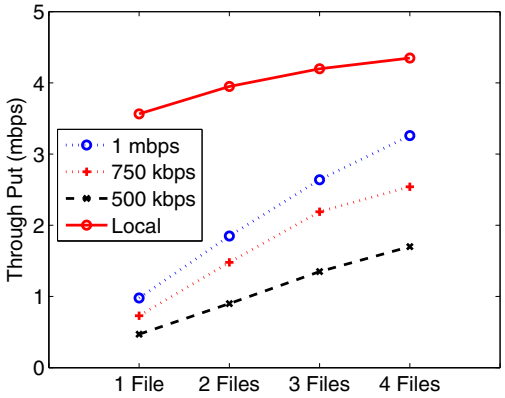


Fig. 5. Parallel Threads for File Read

which the network speed is unconstrained with an average of 3 Mbps, although the advantages tend to diminish along the increase of the file size.

To bridge the performance gap for the sequential read in vUPS read, in the next experiment we use parallel threads for multiple file fetching. Figure 5 shows the result when the number of parallel file fetching grows from one to four. The x-axis represents the number of files fetched in parallel and the y-axis represents the throughput of the system. For the counterpart of the local read, four files are read by four parallel threads from the sdcard of the smartphone.

To read files in parallel from vUPS, the smartphone contacts the main terminal and requests the files. Each file is stored on a random machine selected arbitrarily. The main terminal returns to the smartphone with the web address of the file resources and the smartphone then fetches those files in parallel. Figure 5 shows that the throughput increases with the degree of parallelism.

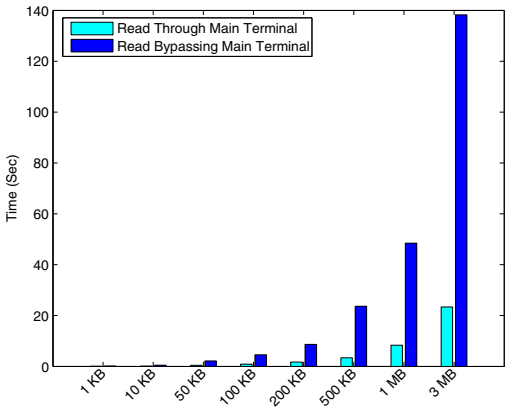


Fig. 6. Performance Gain by Bypassing the Main Terminal

The throughput also increases with the network bandwidth for vUPS when the network bandwidth increases from 500 Kbps to 1 Mbps. When the available network bandwidth is sufficiently large, the gap between the local read and the vUPS read can be significantly reduced.

We have mentioned before that the file operations are handled directly between the requesting machine and the target machine without involving the main terminal in order to relieve the communication bottleneck. To study the performance gain, we compare it to the case when the main terminal fetches the file and then serves it to the smartphone. Figure 6 shows the results averaged over 100 tests. The results confirm that it is beneficial for the user and the system to bypass the main terminal whenever necessary.

5.2 Performance of Metadata Accesses

Metadata is crucial to vUPS to function properly. To measure metadata access performance, we conduct experiments with the *mdtest* benchmark on vUPS. *mdtest* is developed for large scale file systems with script and C, which is not suitable for our RESTful applications. To fit our system, we modify the *mdtest* and implement the simplified version in Java with threading. We replace the file reading/write/edit with Java system calls. In the experiments, we measure the throughput (the number of operations executed per second) for creating 1000 files per terminal from the smartphone.

Figure 7 shows the throughput for file creation. In the experiments, we compare the time for creating files in our smartphone locally and in vUPS. For local file creation in the smartphone, we create 1000, 2000, 3000, and 4000 files using 1, 2, 3, and 4 threads (a thousand files per thread) in the sdcard of the smartphone. To measure the time of file creation in vUPS, we again create 1000, 2000, 3000, and 4000 files using 1, 2, 3, and 4 threads (a thousand files per thread).

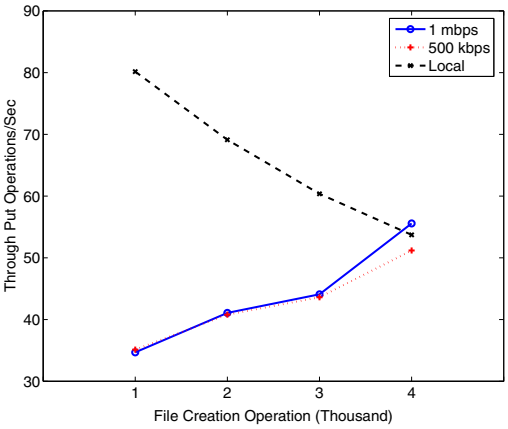


Fig. 7. Throughput of File Creation

Each thread selects one of the devices in the vUPS system. Each thread first contacts the main terminal and requests to create 1000 files in the device.

The result shows that with an increasing number of local file creation requests, the throughput decreases. This is expected because the smartphone thread overhead and the limited file I/O slow down the local file operation, and thus reduce the throughput. On the other hand, for remote devices the performance improves, which is due to the parallel file creation feature in vUPS.

5.3 Network Impact

To study the impact of network bandwidth, we have also run the *mdtest* file creation of 1000 files with varying network speed. Figure 8 shows the response time for 1000 file creation via the smartphone using one thread. The x-axis represents the network speed, while the y-axis represents the system throughput. The figure shows that the network bandwidth does impact the file creation throughput, but the improvement diminishes when the network speed is fast enough. For example, when the network bandwidth is doubled from 5 Mbps to 10 Mbps, the throughput for file creation operation only increases slightly.

To study the impact of network speed on file read operations, we vary the network speed and observe the response time to read a 3 MB file in the smartphone fetched from the main terminal. As expected, Figure 9 shows that the read throughput increases with the increase of bandwidth for vUPS while it remains stable for local read and DropBox. Note that because Dropbox is accessed through the public network, we did not constrain the bandwidth of the connection between the smartphone and the Dropbox.

Table 1 shows the response time for 5 trials with different network speed and file read or creation operations. From ANOVA [29] tests, we may conclude to reject the hypothesis that the operation type and network speed have no

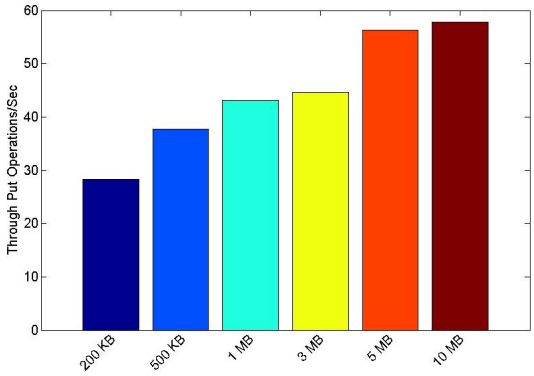


Fig. 8. Network Bandwidth vs File Creation

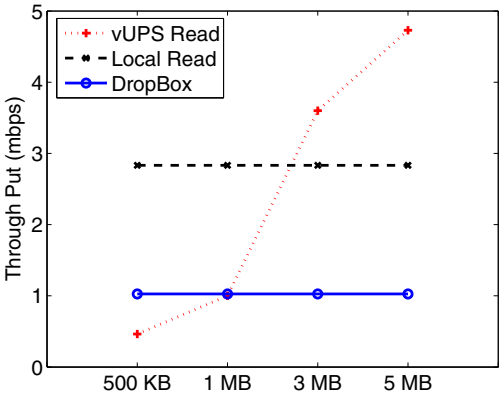


Fig. 9. Network Bandwidth vs File Read

significance on the model. We can also conclude that the interaction between the network speed and the file operation type does exist.

5.4 Comparison to Dropbox

To compare with the DropBox in a more realistic scenario, we have generated 50 files with lambda distribution [37] as discussed in [24] with the size ranging from 1 KB to 512 MB according to that distribution. For DropBox, we synchronize all the files first, then access locally. In vUPS, in case of a cache miss, it downloads the file, otherwise checks the cache and retrieves it unless it is stale. We access these files in random order for 10 to 1000 times.

Figure 10 shows the result. The x-axis represents the number of files fetched randomly from these 50 files. The y-axis represents the response time. In the

Table 1. ANOVA Test for Network and Type of Operation

Operation	500 Kbps	1 Mbps
Read	186	120
Read	204	92
Read	222	164
Read	221	101
Read	228	180
Create	55	58
Create	17	40
Create	37	27
Create	43	32
Create	26	29

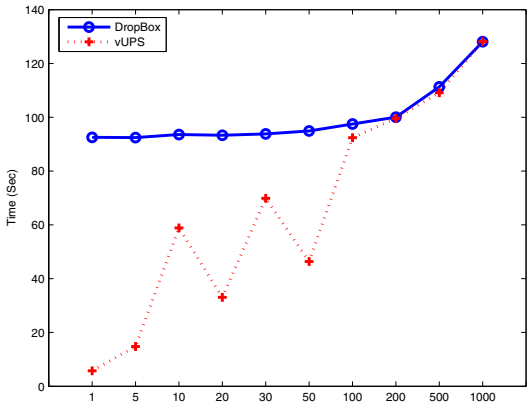


Fig. 10. DropBox vs vUPS

experiment, the DropBox first downloads all the files, and then accesses them randomly. On the other hand, vUPS does not download all the files at first. It downloads the files when being accessed and then caches them for future references. In this way, when the number of accessed file is small, the access time for DropBox is much longer than vUPS as the DropBox first needs to synchronize and download all the files. But when the number of accessed files is larger, the performance of both vUPS and DropBox is similar to each other.

In addition to that, the strong consistency between the replicas introduces synchronization overhead. For example, even a single byte update in dropbox generates 37 KB of network traffic measured by wireshark, where vUPS generates only 1KB of data for the same synchronization.

6 Related Work

Distributed filesystems [38], [28] have been a focus in the field of distributed systems and data replication and consistency have been a major concern for these systems [19].

NFS [38] has been widely used. Coda [39] also proposes offline accesses and replicated access control. Large scale distributed file systems, such as Goolge File System (GFS) [26] and the Hadoop Distributed File System (HDFS) [18,44], have also been well studied for distributed computing. Panache [23] also proposes improvement for parallel accesses. These traditional and research-oriented file systems are designed for local area network and desktop terminals, while in the modern age various mobile devices (e.g., smartphones) play a crucial role for users' data generation and access over public network.

More recently, modern distributed file systems ZZFS [32],Eyo [41], and BlueFS [34] consider the modern portable storage devices such as tablet and smartphones. BlueFS has optimized the data placement policy whose data update was improved by EnsemBlue [35] from a centralized server to peer to peer. ZZFS [32] has specific data placement polices and the consistency policy to avoid conflicts, while Eyo [41] does not guarantee any consistency. PersonalRAID [40] provides device transparency for partial replication which is not suitable for public network as it requires user to move a storage physically between the devices. Cimbiosys [36] provides efficient synchronization with minimized overhead which is not device transparent. But these systems usually have the same replication and consistency policy for all types of data, without distinguishing the very difference between the natures of different types of files.

On the other hand, recent commercial products, such as iCloud [10], Google Drive [9], and Dropbox [5] offer to synchronize data among multiple devices of the same user. But storing users' data on the third party storage potentially risks with security and privacy concerns, while the study [17] shows that the desktop and office computers possessed by an average user is enough to store the data owned by the user.

vUPS has been developed to store the data on the user's own devices without storing data in the public cloud. The API's of vUPS are developed with an aim of the end users who access files from samrtphones and/or other terminals.

7 Conclusion

The pervasive adoption of mobile devices calls for an effective approach to access user's personal files across different devices from anywhere and anytime. While commercial products have offered compelling solutions, users are risking their data security and privacy. In addition, potentially, lots of unnecessary traffic has been wasted during continuous file synchronization. In this work, we have designed and implemented vUPS, a system to transparently and seamlessly integrate a user's personal storage space. A web interface is provided to the user with a global view of the files without involving any third party. Experiments

based on the implemented prototype system show that vUPS can achieve similar user performance when compared to commodity commercial solutions such as DropBox.

References

1. Amazon EC2 outage, http://www.informationweek.com/news/cloud-computing/infrastructure/229402054
2. Amazon S3, http://aws.amazon.com/s3/
3. Cisco Press Release, http://www.rcrwireless.com/article/20110201/
4. Daily Main (April 1, 2012), http://www.dailymail.co.uk/sciencetech/article-2078020/Death-point-shoot-Smartphone-cameras-27-cent-photos.html
5. Drop Box, http://www.dropbox.com/
6. DropBox security breach, http://www.newsy.com/videos/dropbox-security-glitch-leaves-users-accounts-unprotected/
7. EC2 Failure Rate, http://aws.amazon.com/ebs/
8. Google Docs, http://www.docs.google.com
9. Google Drive, https://drive.google.com/
10. iCloud, http://www.apple.com/icloud/
11. International Data Corporation: Press Release (January 28 and February 4, 2010), http://www.idc.com/
12. International Telecommunication Union: Press Release (June 10, 2009), http://www.itu.int
13. Mark Zuckerbergs pictures leaked, http://www.nydailynews.com/news/national/oops-mark-zuckerberg-pictures-leaked-facebook-security-flaw-article-1.988026?localLinksEnabled=false
14. Sky Drive, http://explore.live.com/skydrive
15. SME Storage, http://smestorage.com/
16. Agrawal, N., Bolosky, W.J., Douceur, J.R., Lorch, J.R.: A five-year study of file-system metadata. In: FAST (2007)
17. Bolosky, W.J., Douceur, J.R., Ely, D., Theimer, M.: Feasibility of a serverless distributed file system deployed on an existing set of desktop pcs. In: Sigmetrics (2000)
18. Borthakur, D.: Hdfs architecture: Technical report. In: Apache Software Foundation (2008)
19. Brewer, E.A.: Towards robust distributed systems. In: Principles of Distributed Systems (2000)
20. Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., Terry, D.: Epidemic algorithms for replicated database maintenance. In: PODC (1987)
21. Douceur, J.R., Bolosky, W.J.: A large-scale study of file-system contents. In: Sigmetrics (May 1999)
22. Douceur, J.R., Bolosky, W.J.: A large-scale study of file-system contents. In: SIGMETRICS (2002)
23. Eshel, M., Haskin, R., Hildebrand, D., Naik, M., Schmuck, F., Tewari, R.: Panache: A parallel file system cache for global file access. In: USENIX Conference on File and Storage Technologies (2010)
24. Evans, K.M., Kuenning, G.H.: A study of irregularities in file-size distributions. In: International Symposium on Performance Evaluation of Computer and Telecommunication Systems, San Diego, CA (2002)

25. Fielding, R.T.: Architectural styles and the design of network-based software architectures. PhD Thesis, University of California, Irvine (2000)
26. Ghemawat, S., Gobioff, H., Leung, S.-T.: The google file system. In: Proceedings of the 19th Symposium on Operating Systems Principles (SOSP), NY, USA (2003)
27. Guy, R.G., Heidemann, J.S., Mak, W., Page, Jr., T.W., Popek, G.J., Rothmeier, D.: Implementation of the ficus replicated file system. In: USENIX (March 1990)
28. Howard, J.H., Kazar, M.L., Menees, S.G., Nichols, D.A., Satyanarayanan, M., Sidebotham, R.N., West, M.J.: Scale and performance in a distributed file system. ACM Transactions on Computer Systems (1988)
29. Jain, R.: The art of computer systems performance analysis (1991)
30. Kung, H.T., Robinson, J.T.: On optimistic methods for concurrency control. ACM Transaction on Database Systems (March 1981)
31. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Communicatio of ACM (1978)
32. Mazurek, M.L., Thereska, E., Gunawardena, D., Harper, R., Scott, J.: Zzfs: A hybrid device and cloud file system for spontaneous users. In: FAST (2012)
33. Muthitacharoen, A., Morris, R., Gil, T.M., Chen, B.: Ivy: A read/write peer-to-peer file system. In: OSDI (2002)
34. Nightingale, E.B., Flinn, J.: Energy-efficiency and storage flexibility in the blue file system. In: 6th Conference on Symposium on Opearting Systems Design and Implementation (2004)
35. Peek, D., Flinn, J.: Ensemblue integrating distributed storage and consumer electronics. In: 7th Conference on Symposium on Opearting Systems Design and Implementation (2006)
36. Ramasubramanian, V., Rodeheffer, T.L., Terry, D.B., Walraed-Sullivan, M., Wobber, T., Marshall1, C.C., Vahdat, A.: Cimbiosys: A platform for content-based partial replication. In: Network Systems Design and Implementation (2009)
37. Ramberg, J.S., Schmeiser, B.W.: An approximate method for generating symmetric random variables. Communications of ACM (1974)
38. Sandberg, R., Goldberg, D., Kleiman, S., Walsh, D., Lyon, B.: Design and implementation of the sun network filesystem. In: Summer 1985 USENIX (June 1985)
39. Satyanarayanan, M., Kistler, J.J., Kumar, P., Okasaki, M.E., Siegel, E.H., Steere, D.C.: Coda: A highly available file system for a distributed workstation environment. IEEE Transactions on Computers (April 1990)
40. Sobti, S., Garg, N., Zhangv, C., Yu, X., Krishnamurthy, A., Wang, R.Y.: Personalraid: Mobile storage for distributed and disconnected computers. In: FAST (2012)
41. Strauss, J., Paluska, J.M., Ford, B., Lesniewski-Laas, C., Morris, R., Kaashoek, F.: Eyo: Device-transparent personal storage. In: USENIX Technical Conference (2011)
42. Tanenbaum, A.S.: Distributed systems: Principles and paradigms, 2nd edn.
43. Terry, D.B., Theimer, M.M., Petersen, K., Demer, A.J., Spreitzer, M.J., Hauser, C.H.: Managing update conflicts in bayou, a weakly connected replicated storage system. In: Fifteenth ACM Symposium on Operating Systems Principles (March 1981)
44. White, T.: Hadoop: The definitive guide, 2nd edn.

LineKing: Crowdsourced Line Wait-Time Estimation Using Smartphones

Muhammed Fatih Bulut[1], Yavuz Selim Yilmaz[1], Murat Demirbas[1], Nilgun Ferhatosmanoglu[2], and Hakan Ferhatosmanoglu[3]

[1] University at Buffalo, SUNY, Buffalo, NY 14260, USA
{mbulut,yavuzsel,demirbas}@buffalo.edu
[2] THK University, Ankara 06790, Turkey
nilgunf@thk.edu.tr
[3] Bilkent University, Ankara 06800, Turkey
hakan@cs.bilkent.edu.tr

Abstract. This paper describes the design, implementation and deployment of LineKing (LK), a crowdsourced line wait-time monitoring service. LK consists of a smartphone component (that provides automatic, energy-efficient, and accurate wait-time detection), and a cloud backend (that uses the collected data to provide accurate wait-time estimation). LK is used on a daily basis by hundreds of users to monitor the wait-times of a coffee shop in our university campus. The novel wait-time estimation algorithms deployed at the cloud backend provide mean absolute errors of less than 2-3 minutes.

Keywords: Crowdsourced sensing, Smartphone applications, Wait-time estimation.

1 Introduction

Long and unpredictable line lengths at coffee shops, grocery stores, DMVs, and banks are inconveniences of city life. A webservice that provides real-time estimation of line wait-times would help us make informed choices and improve the quality of our lives. While a line wait-time estimation service may first be regarded as a toy or luxury, it is worth recalling that webservices that we now categorize as necessities (e.g., maps, online shopping, social networks, and mobile internet) have also been perceived as similar initially. Moreover, understanding waiting line has benefits beyond improving the end-user experience because this has been a long standing problem in the operations research area.

Our method to address the line wait-time detection problem is crowdsensing with smartphones. In the very first prototype of our service, we asked users to manually provide line wait-times when they are waiting in line and tried to serve other users with the data input by these. We quickly noticed that this is an extra work for the users and the data arriving from the volunteers is too scarce to serve good results to the queriers. In the later versions of our service, we automated the line wait-time detection by using the localization capabilities of the smartphones in an energy-efficient manner, which we detail in this paper.

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 205–224, 2013.

Line wait-time detection is, however, only one part of the problem. We found that even when our automated line wait-time detection component is returning dozens of readings daily, these readings are still too sparse and non-uniform to provide accurate answers to real-time queries about line wait-times. To address this problem we applied statistical techniques (heuristic regression, exponential smoothing and Holt-Winters method) to the line wait-time data we collect. This allowed us to learn patterns from the current and historical data to provide accurate responses to queries.

Our contributions are as follows:

1. We designed, implemented, and deployed a crowdsourced line wait-time estimation system called LineKing (LK). Our LK apps [1] for Android and iPhone platforms are downloaded by more than 1000 users in our university, and are used on a daily basis by hundreds of users to monitor the line wait-times of a coffee shop in the student union of our university. To the best of our knowledge, LK is the first crowdsourced line wait-time estimation service.
2. As part of LK, we implemented a fully automatic, energy-efficient, and accurate wait-time detection component on Android and iOS platforms. This component uses new domain specific optimizations to achieve better accuracy and performance.
3. For the wait-time estimation problem, we introduced a novel solution based on a constrained nearest-neighbor search in a multi-dimensional space. Then, we improved it by adapting two statistical time-series forecasting methods, namely exponential smoothing and Holt Winters, and demonstrated the strengths and weaknesses of these solutions.
4. We collected and analyzed several months of line wait-time data, which can be basis for other work on similar topics. To extend on our current work, we also discuss the challenges and opportunities for scaling LK to online monitoring of line wait-times over many venues across the world.

Outline of the Rest of the Paper. We discuss related work next and describe the model and limitations of our deployment in Section 3. We present the line wait-time detection component of LK in Section 4 and the line wait-time estimation component in Section 5. Section 6 presents the results from our deployment and experiments. In section 7, we discuss techniques to scale LK to other coffee shops and franchises.

2 Related Work

2.1 Smartphone Sensing

The increasing computational power and sensor capabilities of the smartphones resulted in increasing interest on smartphone sensing [16,9]. In TagSense [21], authors leverage camera, compass, accelerometer and GPS sensors of the phones

[1] `http://ubicomp.cse.buffalo.edu/ubupdates`

to provide an image tagging system. In [2], Bao and Choudhury introduce MoVi that employs smartphones to enable collaborative sensing using videos for recognizing socially interesting events. In [20] Miluzzo et al. utilize smartphone sensors to infer user's status and share on Facebook.

A significant portion of smartphone sensing focuses on location-sensing where both localization and power-aware sensing are explored. In [4], authors examine the human localization in a building using smartphone sensors and randomly placed audio beacons in the building. In [22], authors identify four factors that waste energy: static use of location sensing mechanism, absence of use of other sensors, lack of cooperation among applications, and finally ignoring battery level while sensing. The paper analyzes aforementioned factors and proposes a middleware location sensing framework which adaptively a) toggles between GPS and network based on the accuracy of the providers, b) suppresses the use of location updates based on the context (i.e. user is stationary or moving), c) piggybacks on other applications' location sensing mechanism and d) changes parameters of location updates based on the battery-level. However, that work does not focus on region monitoring and does not take distance into account as a factor for selecting the mode for providing localization. In LK we use distance from the point of interest for toggling the mode for providing the localization.

2.2 Line Wait-Time Estimation

Line wait-time estimation has been explored mostly in the context of Queue Theory [5,17]. Those works assume that examiners have full knowledge of the parameters, i.e. queue discipline, arrival rate, service rate etc. However, in our problem we only have wait-times and the associated timestamps. So queueing theory is not easily applicable for our problem.

Line wait-time estimation is related to some problems in general time series theory where the task is to forecast future data using the previous ones. Number of different techniques have been proposed in the literature ranging from ad hoc methods (i.e. moving average, exponential smoothing) to complex model-based approaches which take trend and seasonality into accounts (i.e. Decomposition, Holt-Winters, ARIMA) [7,19,12]. A major challenge is that general time series analysis depend on data that is uniformly distributed along time. However, our application has non-uniform and initially sparse data that introduce new challenges.

3 Model and Assumptions

We deployed LK at a popular coffee shop at the Student Union of University at Buffalo. Floor plan of the coffee shop is shown in Figure 1. The coffee shop does not have a drive-through. The customers who arrive at the coffee shop join the back of a single FIFO queue. After waiting the line, the customer is served by the staff. There are two service desks and the customer is served by either one of them.

(During low traffic times one of the service desk may close temporarily and only a single service desk is used.) Customers who are served usually leave the coffee shop immediately. However there are some Student Union tables near the service desks and some customers sit there after picking up their coffees. There is a Wi-Fi Access Point (WAP) on the nearby wall of the line to serve customer's need for internet access. The WAP has a range of approximately 50 meters. Our detection system utilizes BSSID of the WAP for wait-time detection.

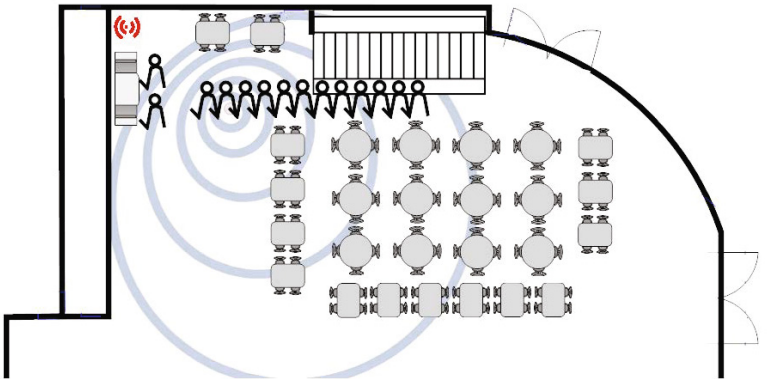


Fig. 1. Coffee Shop Floor Plan

LK aims to estimate the total wait-time of a customer, and does not aim to calculate neither the line length nor the service time. Moreover, our wait-time detection component on the smartphones cannot differentiate between the seated customers and the customers who wait in the line. Therefore, there are two sources of false-positives in our system: 1) a customer who seats after being served, and 2) a customer who takes a look at the coffee shop without waiting in the line. To get a sense of how the wait-time changes over the time, we physically observed the coffee shop continuously for one week. Our observations show that the wait-times almost never fall below 2 minutes (i.e. min. service time) and above 20 minutes. We use this information to eliminate false-positives. Although some false-positives are eliminated this way, customers who sit between 2-to-20 minutes still insert false-positives. Based on our observation sitting customers are the minority with respect to all customers, and our data-analysis techniques manage to filter their data as noise (see Section 5). Later, in Section 7, we explain a way of differentiating seated customers from others to further increase the accuracy of our detection component.

The wait-time detection component on the smartphones can only detect the total wait-time a customer spent in the coffee shop, hence, many parameters remain unknown, such as arrival rate, service rate, service time. This prevents us from having a complete understanding of the line's operational model and introduce many challenges that need to be addressed in wait-time estimation component.

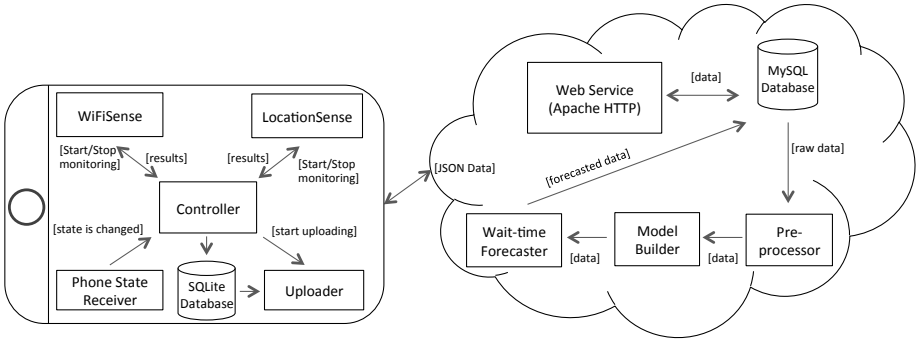


Fig. 2. Overall system architecture. Left: smartphone architecture for wait time detection. Right: cloud architecture for wait time estimation.

4 Smartphone Design

The overall architecture of the system is shown in Figure 2. LK consists of two main components: the client-side component on the smartphone, and the server-side component on the cloud. In this section, we present the client-side component on the smartphone. The server-side component is explained in the next section.

The client-side component includes a controller and three subcomponents: Phone-State-Receiver, Wait-Time-Detection (LocationSense+WiFiSense), and Data-Uploader. The controller is responsible for managing and handling the interactions between these subcomponents. We explain each subcomponent in detail next.

Remark: In the following sections, we describe the smartphone component designed for the Android Operating System [1]. Due to the development limitations that iOS imposes, some of the features below are not available for iOS [11]. We will refer to such features in the text to distinguish those parts.

4.1 Phone State Receiver

This component serves as a notification center for the application. Android provides a notification service to let apps know about various events occurring on the device, such as Boot, Reboot, Proximity Alert, Wi-Fi Signal Strength Change etc. iOS also has similar functionalities such as Significant Location Change action. These notifications enable apps to take action based on relevant events. We exploit this notification service in order to improve the wait-time-detection subsystem.

The Phone-State-Receiver subsystem has three different receivers which are Boot Receiver, Wi-Fi State Receiver and Proximity Alert Receiver. In Android, receivers work as follows: First, each receiver registers itself to listen specific events occurring on the device. Whenever the registered action happens, the

operating system broadcasts a special object, i.e. an Intent, and delivers the event specific information to all registered receivers. We utilize this mechanism to monitor various relevant events for our application. For example, the Wi-Fi State Receiver gets notified when the state of the Wi-Fi connection is changed: So if the user turns the Wi-Fi off, this receiver fires at the Controller to stop the Wi-Fi Tracking Service if it is running. Another example is the Proximity Alert Receiver which notifies our app of entering and exiting the coffee shop. We explain this alert in detail next.

How does proximity alert work?
Proximity alert is a service provided by the Android OS (Region Monitoring for iOS is also present) that periodically checks the location of the device and fires alerts for the entering and exiting events for a specified geo-fenced region. The programmer can set the proximity alert by providing a location (i.e., latitude and longitude) and a radius, which represents a circle around the location. On the other hand, the device also has a location and its location has an accuracy. Hence, while one circle denotes the geo-fence that the application tracks, another circle denotes the device's location. When these two circles intersect *entered* event is fired, and when they are separated *exited* event is fired.

A natural way to detect wait-time is to set proximity alert for the coffee shop and use entered and exited events to get notified. However, continuous use of proximity alert drains the battery quickly. Figure 3 shows the battery consumption of an Android device which registers for only one location compared to another Android device with no proximity alert. Even if we don't move the device (same location), the former drained the battery in 18 hours compared to almost 75% remaining battery level on the latter device.

Due to the costs of proximity alert (especially in Android), we do not use proximity alert directly. As we explain in the next section, our system sets a proximity alert only if the user is close to the targeted location. Otherwise, our system adaptively checks if the user is making progress towards the coffee shop by monitoring the distance.

4.2 Wait-Time Detection

The wait-time detection subsystem uses the device location and the BSSID of the Wi-Fi Access Point (WAP) to detect the user's presence at the coffee shop. Due to the high energy cost of continuous sensing, we adopt a hint-based approach to initiate sensing. We utilize the following two hints: First, we assume that if a user opens the app to check the line wait-time, then she is a potential candidate to visit the coffee shop soon. Second, we utilize the user's coarse-grained location and start/stop monitoring if the user is close to the coffee shops. By using these two hints LK achieves energy-efficient monitoring.

LK employs two alternative methods to detect the wait-time at a coffee shop: Location-sensing and WiFi-sensing. Both methods are orthogonal and return results with similar accuracies, which makes the two methods replaceable. If the coffee shop does not have a WAP (or the WAP is not learned/validated yet),

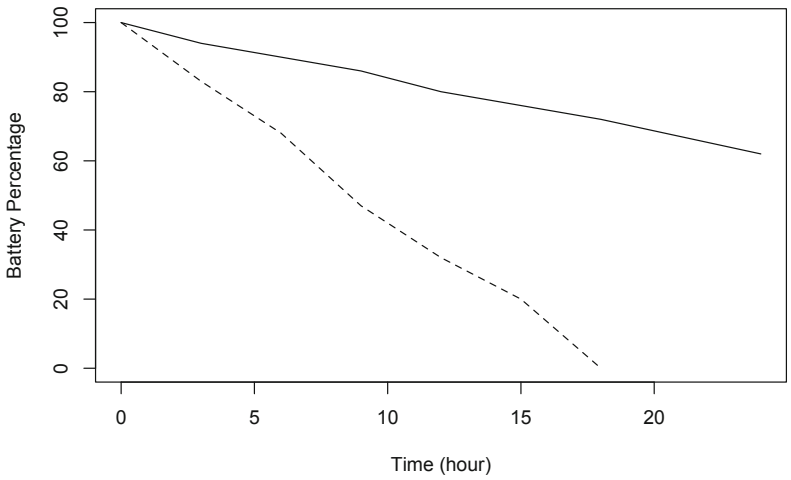


Fig. 3. The effect of using proximity alert continuously. The dashed line represents the battery level of a device which registers for one proximity alert, and the solid line represents a device which registers for none.

then the Location-sensing can be used. If the coffee shop is inside a big mall where the Location-sensing does not work accurately, then the WiFi-sensing is more advantageous. In our deployment we used both methods, and we provide results from both in the experiments section. We describe these two techniques next.

Location-Sensing. Once the smartphone component has a hint that the user may go to a coffee shop, it dispatches a new job and starts monitoring the user's location. First the distance between the coffee shop and user's current location is calculated. If the user is close enough (i.e. $R < 100$ meters) to the targeted location, then the app sets a proximity alert to detect the timestamp of entering to and leaving from the coffee shop. However, if the user is not close enough ($R >= 100$ meters), then the app schedules an alarm to check the location again on the estimated arrival time of the user. The estimated arrival time is calculated based on the user's current speed and her distance to the coffee shop. Before scheduling further alerts, our app expects user to make some progress towards the coffee shop. If the user does not make any progress towards the targeted location n consequtive times, then the job is cancelled. If there is a progress towards the targeted location, then the proximity alert lets our app know at time t_1 where user enters to and at time t_2 where user leaves from the coffee shop. Having these two timestamps, wait-time is just the difference of $t_2 - t_1 - \epsilon$ where ϵ is the mean error which is accounted for the accuracy of this method (around 1.5 minutes).

Location provider selection in smartphones:
Smartphones provide various ways to obtain user's location. GPS, cell towers, and WAPs are the main ones. Aside from these, there are two other mechanism to learn location: last-known location and passive location learning—a way of getting location whenever another application requests it. The location provided by these methods has its accuracy and timestamp on it. In Android, an application for tracking the location is free to choose among these methods. However, given the different sensing costs of these methods in terms of energy and time, we use a dynamic and adaptive selection of the location providers.

Since we are interested in the user's proximity to the coffee shop, distance is an important parameter for our system. Hence, using the user's distance from the targeted location, our app dynamically selects to use cell tower, WAP or GPS as a provider (see Figure 4). Android combines and uses cell tower and WAP locations as the network location and differentiate it from the GPS location. In Android, our app first looks for the last-known location and calculate the user's distance by taking accuracy and timestamp of the location into account. If the Coffee Shop is very far away or the location is recent and accurate, then we use the last-known location. Otherwise, we take the network location which in general has a better accuracy. If it doesn't satisfy the requirements too, then we take a more conservative approach and learn the location from GPS which in general has a good accuracy of 10 meters. Note that last-known location and network location is mostly available and they are not costly in terms of the battery. On the other hand GPS is costly and can easily drain the device's battery if over-used.

Wi-Fi Sensing. Our second method for wait-time detection is to leverage the WAP in the coffee shop. Nowadays, most of the coffee shops have Wi-Fi to provide their customers easy and fast access to internet. Moreover, these WAPs generate beacons to broadcast their existence within the radius of 50-100 meters. And each beacon associates with a unique BSSID.

Once our system has a hint that the user will potentially visit the coffee shop, it then starts to monitor Wi-Fi beacons periodically to detect entering to and exiting from the targeted location. With the help of scanning ability of WAPs without connecting to them (provided on the Android platform), our system tracks Wi-Fi beacons easily with little energy consumption. Having the Wi-Fi scan results available, wait-time calculation is just the process of taking difference of $t_2 - t_1 - \epsilon$ where t_1 is the time we start and t_2 is the time when we stop to see Wi-Fi beacons of the WAP. Note that ϵ is the mean error for this method (around 1 minutes, accounting for the scanning period).

4.3 Data Uploader

After completing the wait-time-detection, the smartphone component tries to upload the resulting data to the cloud as an input to our wait-time estimation system. Since the data is reasonably small, the uploading process is mostly successful in real time. However, due to the status of the device or connection,

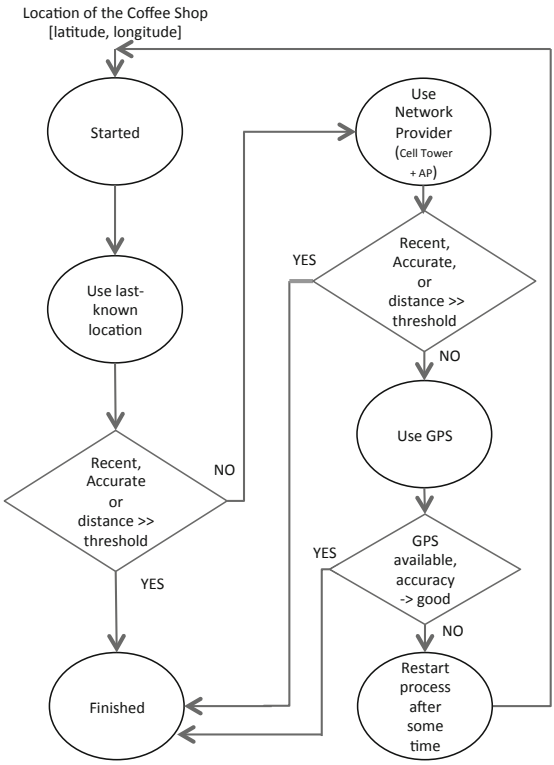


Fig. 4. Selecting among location providers

sometimes it is not possible to transmit data immediately. To handle this case, we have a data uploader subsystem. The data uploader is responsible for transmitting the pending wait-time detection data whenever the Phone-State-Receiver notifies the Controller about the availability of a Wi-Fi or GSM data connection.

Since a failed data transfer costs some energy, the data uploader uses some simple heuristics to increase the upload success rate. We assume that the device is charged mostly when the user is at home or office where she has a reasonably fast and reliable data connection, which is most of the time a Wi-Fi connection. Therefore, the data uploader is triggered when the device is connected to a power outlet to leverage this efficient and reliable connection. Under some circumstances, even if the device is being connected to a power outlet, it may not have such data connection available. If so, then the data uploader periodically (once an hour) checks for a data connection.

Data uploader stores the pending transfers inside a database that resides on the device. The data is sent to server as a JSON object using HTTP POST. Once the data is successfully sent, which is confirmed by a response from the server side, then the Data uploader clears up the database in order to save some storage on the device.

5 Wait-Time Estimation

In this section, we present the wait-time estimation component of LK. This component resides on the server-side (hosted on AWS EC2 cloud for scalability) and consists of four main components: Web service, Pre-processor, Model-builder and Wait-time forecaster. The web service serves as the interface between smartphones and the back-end. It accepts wait-times collected from the smartphones and provides wait-time estimations for the querying smartphones. Data collected from web service is fed into the pre-processing module which is responsible mainly for removing outliers and smoothing the data. After pre-processing, model builder builds a model from all the collected data. Lastly, the wait-time forecaster module uses the model and estimates the future wait-times. Below we describe the data that we use for analysis, then we explain wait-time estimation in details.

Data: For our analysis, we used the 8 weeks of collected data (CD) between 2012-02-27 and 2012-04-29 from 8am to 5pm [2]. CD consists of the wait-time detections that the LK app on the smartphones generated and transferred to the back-end. This data is non-uniformly distributed along time, and is initially sparse over the period it has been collected. The sparseness of the data gradually decreases as the popularity of the application increases.

The raw CD contains outliers due to false-positives: some customers sit in the coffee shop after being served, and some leave the coffee shop without being served. Thus, CD needs smoothing and outlier removal methods to filter out the samples that do not provide direct information of wait-times. We utilize distance-based outlier detection method defined in [14] to enable more accurate modeling. Also, during preprocessing, we remove wait-times that are smaller than 2 minutes (which is less than the observed min. service time) and larger than 20 minutes (which is larger than the observed max. wait-time). As a result there are total of 1782 data points in our CD dataset.

Separate from CD, we also manually collected one week of Observed Data (OD) by physically observing and noting the wait-time in the coffee shop. OD consists of wait-time data collected with equal intervals (every 10 minutes) between 8am-5pm, and provides an accurate state of the line wait-times (without false-positives) for that week. Of course it is tedious to collect OD, and we cannot expect to obtain OD for all the businesses added to LK. Since we want our LK service to be scalable and bootstrap itself from the beginning, the wait-time estimation techniques we developed do not rely on OD. We use OD only to observe how wait-times changes along the time and to extract max. and min. wait-times.

Wait-Time Estimation Problem: The problem is to estimate the line wait-time for any arriving query by using CD. Although the queries can be for anytime (past, now, future); we expect real-time querying for the current time (e.g., 5-10 minutes in to the future) to be most useful. Hence, the wait-time estimation

[2] A week is excluded due to spring break.

models need to access the most up-to-date information in CD. Wait-times usually depend on i) the time of the day, ii) weekday vs. weekend, and iii) seasonality depending on the nature of the business. For our specific coffee shop, there is less traffic in off-school days and weekends, and slightly more traffic in certain times of a day. An estimation method should capture all of these variables accurately.

The theory of time-series estimation has been usually based on regular uniform time-series that contain enough samples. In our case, the data is neither complete nor uniform. Therefore, a general theory of time-series is not directly applicable on CD. However, as the popularity of the application increases and by employing techniques for filling missing data, we can overcome this challenge and build robust models to estimate wait-times. To achieve this, we developed two estimation approaches. Our two estimation approaches represent a spectrum from a fast heuristic to a time-series model. Both approaches are designed to handle insufficient data and adapt/improve as more data is collected.

Our first approach is a Nearest neighbor estimation (NNE) based on constrained nearest-neighbor search in a multi-dimensional space. This approach is dynamic and works well with non-uniform and sparse CD. In the second approach, we improve NNE by building time-series model on CD using the previous history of wait-times. We show that both approaches provide considerably accurate estimations.

In Section 5.1 we explain our nearest neighbor estimation technique. In Section 5.2, we present the model-based time series estimation and finally in Section 5.3, we compare the estimation (forecasting) capabilities of the mechanisms.

Evaluation: We evaluate the approaches using their resulting Mean Absolute Error (MAE). Given a set of n wait-times: $y_1, y_2, ..., y_n$ and their estimated values: $f_1, f_2, ..., f_n$, MAE is defined in Equation 1.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i| \tag{1}$$

5.1 NNE: Nearest Neighbor Estimation

The main idea in this method is to predict the queried values using the previous history of wait-times based on their similarity of values. To this end we identify k nearest neighbor points (k-NN) for the query, where similarity is defined with respect to the estimation potential. The key here is to design a similarity (neighborness) function that optimizes the estimation error for the query.

In order to realize this method, we define every data point with 3 dimensions: week, day and day-interval, $[w, d, i]$. Each data is associated by a vector $[w_i, d_i, i_i]$, where w_i stands for the week of the year and is from the domain [1,52], d_i stands for day of the week and from the domain [1,7], i_i stands for interval of the day and is from the domain [1,54] (there are 54 intervals of 10 minutes between 8am and 5pm). We use weighted L_{ij} to denote the dissimilarity measure between two vectors and define it as the weighted sum of the absolute differences between each dimension;

$$L_{ij} = \alpha(|w_i - w_j|) + \beta(|d_i - d_j|) + \gamma(|i_i - i_j|) \quad (2)$$

Hence, the problem is deduced to find the optimal values for α, β and γ. Below we explain our regression-based optimization method to optimize these values.

Regression-Based Optimization. In statistics, it is a common practice to use regression to understand the relationship between regressand and regressors. For our case, we want to quantify the relation between the wait-time (v_i) and the data vector ($[w_i, d_i, i_i]$). Therefore, we first assume that wait-time is linearly dependent to each dimension of the data vector as in Equation 3. And then we utilize the labeled data points (previous history of wait-times) and assign the weights that optimize the regression function for the labeled data.

$$v_i = \alpha w_i + \beta d_i + \gamma i_i \quad (3)$$

We use linear regression to optimize these weights. Table 1 shows the normalized dissimilarity weights for our 8 weeks of data. Results indicate that, for an estimation interval, the nearest intervals and the intervals from the nearest days have higher similarity than the intervals from previous weeks. This roughly means that the importance of the previous week's data decreases as the time passes. This provides a dynamic way of selecting closest week's data for our method.

Table 1. Weights for dissimilarity measurement using regression

Week (α)	**Day (β)**	**Interval (γ)**
0.991	0.130	0.032

K-NN Estimation. After finding the weights (α, β, γ), similar to the k-nearest neighbor algorithm in machine learning [6], our aim is to find the k nearest neighbors for the queried data point. For this purpose, we first calculate the distance of the query to each of the labeled data points. And then, we find the minimum distanced $k = 5$ data points and calculate the average of their wait-times as the estimated value. First row of Table 2 shows the modeling error of NNE. Note that results are in terms of seconds. Same applies to all following results.

Table 2. Modeling error for the models

Model	**MAE**
NNE	234
Exp. Smoothing	34
Holt Winters	55

5.2 Model-Based Estimation

In this section, we explain how we apply time series theory to wait-time estimation problem as an improvement to our NNE method. As we stated earlier, time series analysis usually depends on uniform and equally spaced data. However, CD does not fully convey these features. Besides it has outliers that makes modeling more difficult. In this section, we present solutions to overcome these shortcomings on CD. We first present how we generate uniform, equally-spaced time series data which we call enhanced collected data (e-CD). Second, we provide the analysis of e-CD. Finally, we fit the data to the time series forecasting models and provide results to evaluate their performance while comparing with our previous approach.

Missing Data Problem on CD. In a typical data collection process, missing data can occur for a variety of reasons including system failures, communication failures etc. In our case, we have missing data because in some intervals either there was no users using our app in the coffee shop or the existing users were unable to upload detected wait-times to our server yet. Although, we expect these behaviors to minimize as more users use our app, until then we need to handle these missing data and provide accurate wait-time estimation.

There are variety of ways to handle missing data; imputation, partial deletion, interpolation and regression are just some of the methods [18]. We adopt a regression-based estimation for filling and constructing equally-spaced (10 minutes of intervals) data. Specifically, we adopt our nearest neighbor estimation (NNE) method that we defined in the previous section for handling missing data. Since we cover most of the details of the method in previous section, we will not repeat the same phenomena here. Basically, we fill the missing interval using the resulting regression model which is constructed by using previous history of wait-times up to the queried interval. For the intervals that already have data, we simply take the average of the collected data as the representative of that interval. In order to deal with outliers we apply a two-sided moving average smoothing to the data. After these processes we finally constructed equally-spaced (10 minutes), uniform time series; e-CD.

Analysis of e-CD. In this section we analyze the enhanced collected data (e-CD) in details. This would enlighten our selection of forecasting models in the next section. Figure 5(a) and (b) illustrate some statistical diagnostics of time series. The bell-shaped histogram plot and the wait-times being close to the straight line in the normal probability plot (although a little bit skewed left for the higher values) indicate that the wait-times follow a normal distribution. Note that time-series models are more suitable for normally distributed data and e-CD exhibits this behavior [3].

Traditionally, time-series data considered to be composed of trend, seasonality, cyclical fluctuation and the random effect. Figure 5(c) and (d) shows trend and seasonal components of e-CD. As can be seen from the Figure 5(c), there

is an increasing trend for the wait-time which we believe due to the approaching summer. Figure 5(d) shows the seasonality component of e-CD. It clearly exhibits seasonality and validate the selection of a model which has seasonality component.

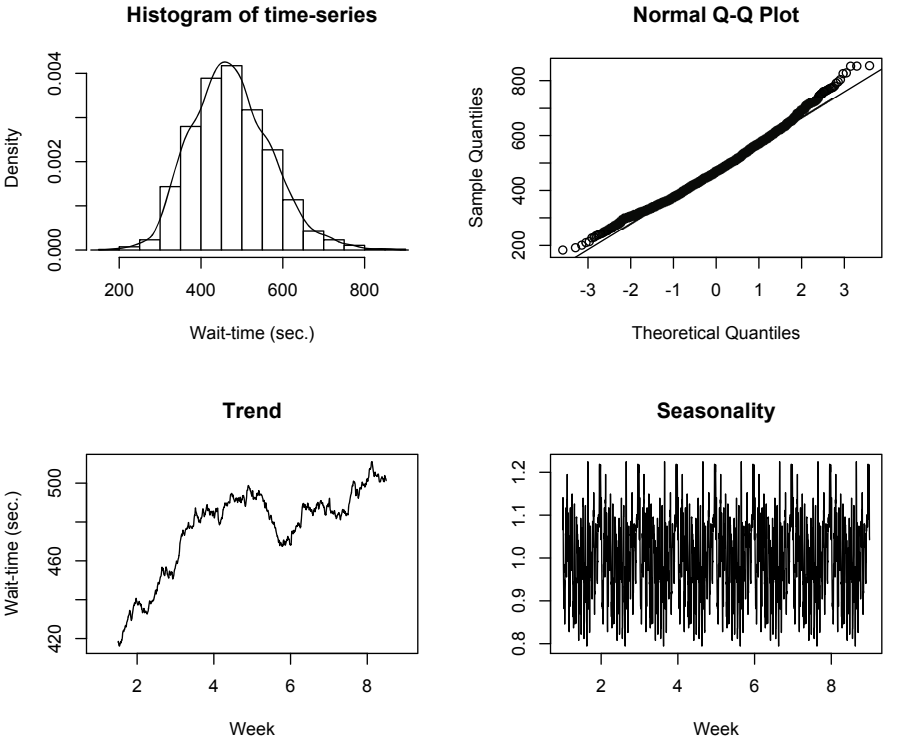


Fig. 5. Statistical plots of e-CD. (a) Histogram of wait-times (b) Normal Q-Q Plot (c) Trend component of time series (d) Seasonality component of time series.

Modeling e-CD. As explained in the previous section, e-CD exhibits trend and seasonality which motivate us to use a model which takes these into accounts. In addition, it should be as light-weight as possible and can be incrementally updated as more data comes. This encourages us to use Holt-Winters forecasting method which is a widely used time-series forecasting model based on exponential smoothing [10]. We experimentally select multiplicative seasonal model and choose data(α_{hw}), trend(β_{hw}) and seasonal(γ_{hw}) smoothing factors dynamically as more data comes. We've also considered other forecasting methods such as ARIMA (Autoregressive integrated moving average) and exponential smoothing. However, we observed that ARIMA incurs significant computational costs, and therefore we opt-out it and compare Holt Winters with the ad hoc exponential smoothing. Second and third rows of Table 2 shows the modeling error

of Holt Winters and exponential smoothing. It is clear that the both models fits perfectly to our e-CD in comparison to our initial NNE method.

5.3 Comparison of Models

In this section, we compare the models built in the previous two sections in terms of their forecasting capabilities. Table 3 shows the forecasting errors of the models for the last two weeks of our experiment. It is expected that forecasting error will be higher than the modeling error as the modeling takes all data into account from the beginning. On the other hand, forecasting only uses the previous data from the queried one. As shown in the Table 3 Holt Winters outperforms NNE in a significant scale. Figure 6 shows the weekly MAE for each method. Holt Winters and exponential smoothing outscores the NNE for all of the weeks consistently. We believe this is due to the fact that wait-time changes steadily rather than immediately over time and both Holt Winters and exponential smoothing account this fact in its formulation by steadily changing the estimation with new incoming data. Moreover, Holt Winters and exponential smoothing exhibits similar errors where Holt Winter beats exponential smoothing in small margins for most of the weeks. We believe that margin will increase as more data comes and as the data exhibits more trend and seasonality. Figure 7 shows the collected data (including replaced missing values) and forecasted values for the last two days of the last week of our experiment for Holt Winters method. Forecasted values nicely fitted to the actual values. Our current wait-time estimation model based on Holt Winters and updated as more data accumulates.

Table 3. Forecasting error for the last two weeks

Model	MAE
NNE	227
Exp. Smoothing	156
Holt Winters	155

6 Deployment

Section 5 presented analysis and experiments of the collected data for building an accurate wait-time estimation model. In this section we present other information from our deployment. We implemented native Android and iPhone apps for LK and made this available at the corresponding appstores for free. While Android app is written in Java using Eclipse and Android SDK, iPhone app is written in Objective-C using Xcode and iOS SDK. For the sake of scalability, we hosted back-end at AWS EC2. A screenshot of the application is shown in Figure 8. It shows the current wait-time and a graph showcasing the past (left) and future (right) estimates of wait-times. We advertised LK through handing out fliers and putting ads on Facebook pages of various Student Clubs. As of writing this paper, LK was downloaded by more than 1000 users in our campus. We received lots of positive feedbacks about the accuracy of estimation from the users.

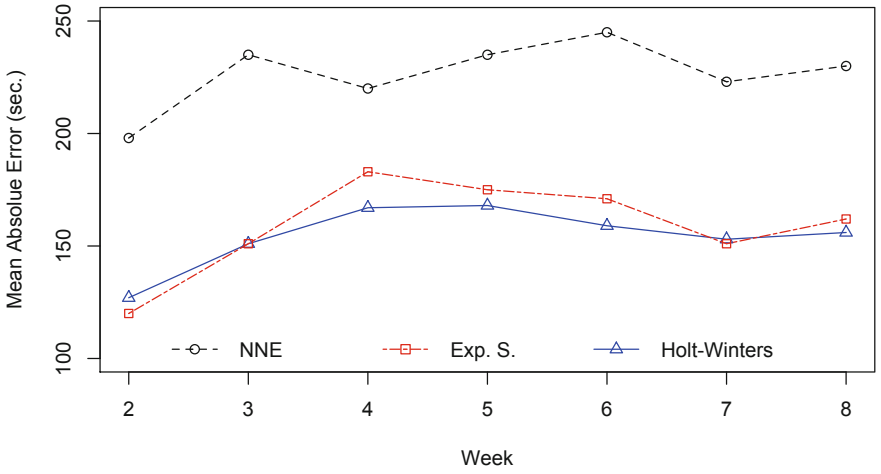


Fig. 6. Weekly forecasting errors for models

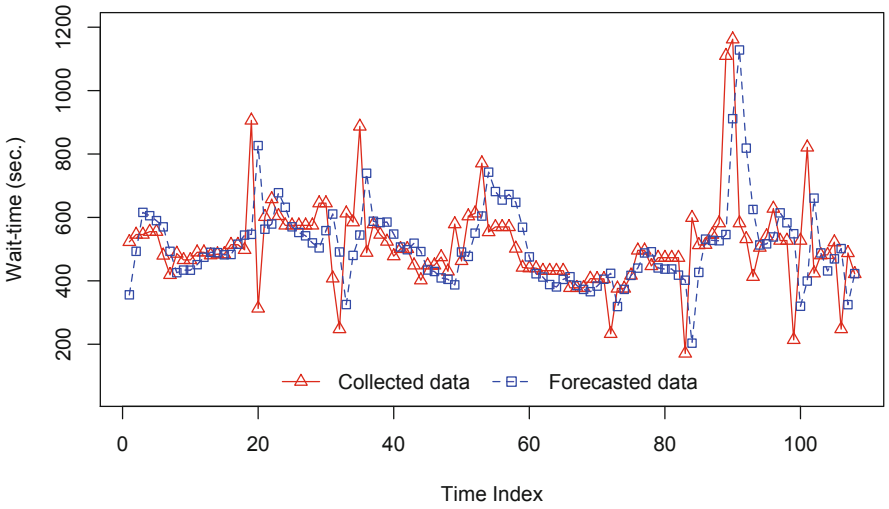


Fig. 7. Collected vs. forecasted data for the last two days of the last week for Holt Winters method. Time Index spans 10 minutes of interval between 8am and 5pm for two days.

Since the iOS platform does not provide a lot of development flexibility, we were unable to implement Wi-Fi sensing for wait-time detection and ended-up using only the location-sensing based solution on the iPhone platform. This was not an issue for the Android platform and both Wi-Fi sensing and location-sensing solutions are fully implemented on the Android. On the other hand, we found that the iOS platform had its own advantages: it was easier to implement a robust location-sensing on the iPhone than on the Android platform.

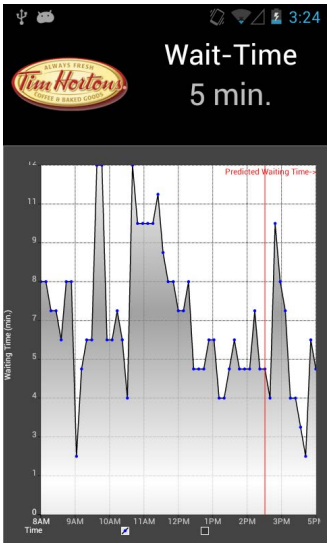


Fig. 8. A screenshot from Android app

As we explained in Section 4, LK receives wait-time detection from two sources of information, i.e. location and WAP. Our 8 weeks of CD show that 65% of incoming data is received from location-sensing (iPhone + Android), and the remaining 35% is received from Wi-Fi based sensing (only Android). Figure 9 shows the number of data points for the weeks we used for analysis, except that data is not preprocessed and it is for all day.

7 Discussion

While we presented LK's deployment for one coffee shop, we believe LK's deployment can be extended for other coffee-shops and businesses such as Post Offices, Banks and DMVs. To add a new business to the LK, we only require the geographical locations, i.e. latitude and longitude, of the business. After a business is added, LK immediately starts receiving line wait-time data from the users visiting that business. Depending on the number of LK users visiting the business, it may take time for LK to construct a model and start providing accurate wait-time estimations for the business. To speed up this process, a business added to LK may manually provide wait times for a week, or offer promotions and coupons for users who install the LK app and check-in frequently. Below, we include more details on how to scale LK to a large set of locations.

7.1 Automated Learning of BSSID

In our reported deployment we manually learned the BSSID of the WAP in the coffee shop. However, in order to scale LK to other locations quickly, we can

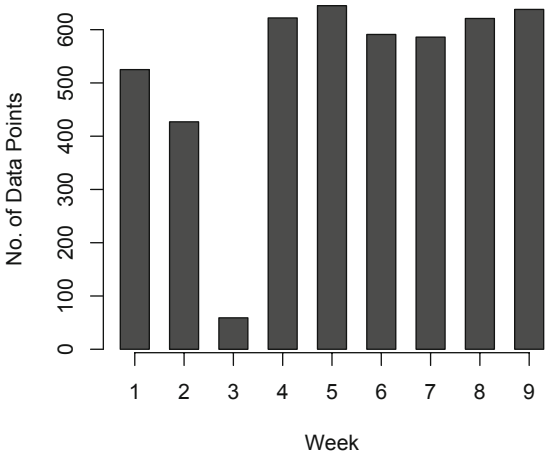


Fig. 9. Incoming number of data points (for all day). Week 3 is spring break and excluded from the wait-time analysis.

automate this process as follows. Initially when the BSSID of the WAP in a business is still unknown, LK relies on just the Location sensing mechanism for wait-time detection. During this phase, LK app instances scan for the available WAPs in that business location and upload these to the LK servers. Learning and validating the BSSID of a business involves recurring observations of the same BSSID by different users at different times. After the BSSID of the business is learned, LK starts accepting line wait-time detections from that business via WAP as well. This increases the data collected from that business and shortens the period for constructing an accurate wait-estimation model.

7.2 Integrating LK with Social Networks

We plan to use social network services and APIs to quickly scale LK for line wait-time monitoring of businesses nationwide and worldwide. For example, we will obtain the geographical locations of new businesses to add to LK by using the Foursquare [8] Venue API (which does not even require a login to Foursquare). We also plan to integrate/embed LK as an extension to the existing popular location-based services such as Facebook, Foursquare and Google Places.

7.3 Improving Wait-Time Detection

As we explained in previous sections, wait-time detection component of LK cannot differentiate between seated customers and the customers waiting in line. In our deployment, majority of the customers leave the coffee shop immediately, therefore, false-positives do not constitute a problem. However, until enough user base is formed, it is possible that wait-time estimation at new businesses might suffer from these false-positives. In order to eliminate these false-positives

in wait-time detection, we will try to distinguish between seated customers and waiting customers by employing the state-of-the-art activity recognition techniques [15,13]. These techniques use the accelerometers in the smartphones to differentiate between different behaviors, including sitting and standing.

8 Conclusion

We described the design, implementation and deployment of LK, a crowdsourced line wait-time monitoring service. LK consists of two main parts: smartphone and cloud back-end components. Smartphone component provides automatic, energy efficient and accurate wait-time detection by using domain specific optimizations for both Android and iOS. And cloud back-end provides accurate wait-time estimation based on collected data from smartphones. In wait-time estimation, we introduced a novel solution based on a constrained nearest-neighbor search in a multi-dimensional space. We then improve it by adapting two time-series forecasting methods namely exponential smoothing and Holt Winters. Our experiments show that, we managed to reduce the mean absolute error of our service to be less than 2-3 minutes. In our future work, we will add new businesses to LK and try to scale our wait-time estimation service to a nationwide deployment.

References

1. Android SDK, `http://developer.android.com`
2. Bao, X., Choudhury, R.R.: Movi: mobile phone based video highlights via collaborative sensing. In: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, MobiSys 2010, pp. 357–370. ACM, New York (2010)
3. Brockwell, P.J., Davis, R.A.: Time series: theory and methods. Springer-Verlag New York, Inc., New York (1986)
4. Constandache, I., Bao, X., Azizyan, M., Choudhury, R.R.: Did you see bob?: human localization using mobile phones. In: Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking, MobiCom 2010, pp. 149–160. ACM, New York (2010)
5. Cooper, R.B.: Introduction to Queueing Theory, 2nd edn. North-Holland, New York (1981)
6. Dasarathy, B.V.: Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques. IEEE Computer Society Press, Los Alamitos (1991)
7. Faloutsos, C.: Mining time series data. In: SBBD, pp. 4–5 (2005)
8. Foursquare venues platform, `https://developer.foursquare.com/overview/venues`
9. Ganti, R.K., Ye, F., Lei, H.: Mobile crowdsensing: Current state and future challenges. IEEE Communications Magazine 49(11), 32–39 (2011)
10. Holt, C.C.: Forecasting seasonals and trends by exponentially weighted moving averages. International Journal of Forecasting 20(1), 5–10 (2004)
11. iOS SDK, `https://developer.apple.com`

12. Kalpakis, K., Gada, D., Puttagunta, V.: Distance measures for effective clustering of arima time-series. In: Proceedings of the 2001 IEEE International Conference on Data Mining, ICDM 2001, pp. 273–280. IEEE Computer Society, Washington, DC (2001)
13. Keally, M., Zhou, G., Xing, G., Wu, J., Pyles, A.: Pbn: towards practical activity recognition using smartphone-based body sensor networks. In: Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems, SenSys 2011, pp. 246–259. ACM, New York (2011)
14. Knorr, E.M., Ng, R.T.: A unified approach for mining outliers. In: Proceedings of the 1997 Conference of the Centre for Advanced Studies on Collaborative Research, p. 11. IBM Press (1997)
15. Kwapisz, J.R., Weiss, G.M., Moore, S.A.: Activity recognition using cell phone accelerometers. SIGKDD Explor. Newsl. 12(2), 74–82 (2011)
16. Lane, N.D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., Campbell, A.T.: A survey of mobile phone sensing. Comm. Mag. 48, 140–150 (2010)
17. Lindley, D.V.: The theory of queues with a single server. Mathematical Proceedings of the Cambridge Philosophical Society, vol. 48(02), p. 277 (1952)
18. Little, R.J.A., Rubin, D.B.: Statistical analysis with missing data. John Wiley & Sons, Inc., New York (1986)
19. Maimon, O., Rokach, L. (eds.): Data Mining and Knowledge Discovery Handbook, 2nd edn. Springer (2010)
20. Miluzzo, E., Lane, N.D., Eisenman, S.B., Campbell, A.T.: CenceMe – Injecting Sensing Presence into Social Networking Applications. In: Kortuem, G., Finney, J., Lea, R., Sundramoorthy, V. (eds.) EuroSSC 2007. LNCS, vol. 4793, pp. 1–28. Springer, Heidelberg (2007)
21. Qin, C., Bao, X., Choudhury, R.R., Nelakuditi, S.: Tagsense: a smartphone-based approach to automatic image tagging. In: Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services, MobiSys 2011, pp. 1–14. ACM, New York (2011)
22. Zhuang, Z., Kim, K.-H., Singh, J.P.: Improving energy efficiency of location sensing on smartphones. In: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, MobiSys 2010, pp. 315–330 (2010)

FlexiSketch: A Mobile Sketching Tool for Software Modeling

Dustin Wüest, Norbert Seyff, and Martin Glinz

University of Zurich, Department of Informatics, RERG,
Binzmühlestr. 14, 8050 Zurich, Switzerland
{wueest,seyff,glinz}@ifi.uzh.ch

Abstract. Although most software engineers have access to various modeling tools, they often use paper and pencil to sketch ideas and to support modeling activities. This is particularly true when they are working in the field, for example gathering requirements from stakeholders. Sketches documented on paper very often need to be re-modeled in order to allow further processing – an error-prone and time-consuming task. The aim of our work is to better integrate these early sketching and modeling activities into the overall software engineering process. We have prototyped FlexiSketch, a mobile application that supports freeform, flexible, in-situ modeling and allows software engineers to annotate their models. Apart from the application and the underlying conceptual solution we also present the results of initial experiments. Those suggest that the tool supports freeform sketching similar to paper and pencil, and that practitioners would be willing to use a tool like FlexiSketch in their daily work.

Keywords: flexible, sketch-based, modeling, software engineering.

1 Introduction

Early software engineering (SE) activities include gathering and documenting needs that stakeholders of a future software product have. Also, first design sketches of the future software system are created. These tasks often ask for creativity. Diagram-like sketches (consisting of nodes and edges) can help to depict current systems and communicate ideas in a simplified way.

Engineers and stakeholders mostly prefer to use paper and pencil, whiteboards, and flip charts for sketching in early SE phases [1,2]. This is mainly due to two reasons: First, engineers and stakeholders meet in various places. Paper and pencil, or flip charts, are available anywhere and are instantly ready for use. Second, they are easy to use. They allow for informal, unrestricted sketches. In contrast, most SE modeling tools follow a more formal modeling approach [3] and require an engineer to use a specific modeling language and syntax.

Once relevant ideas are documented on paper, the question is how to store, distribute, and make the information amenable for further processing. One option is to take photographs of sketches and later, back in the office, re-model

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 225–244, 2013.

the information manually with the help of a software modeling tool. This media break is error-prone and the re-modeling task time-consuming. Information may get lost or be interpreted in a wrong way. This is particularly a risk when another person is responsible for the re-modeling task or information is transcribed after some time has passed by.

To avoid the need for re-modeling information manually, we propose that software engineers use lightweight software tools right from the beginning of a project. Such tools must have a similar availability to that of paper and pencil-based approaches and allow for freeform sketching. In our current research, we focus on multi-touch mobile devices such as tablet computers to support these sketching activities. Today, such devices are widespread, ad-hoc available, and have sufficient computing power. Therefore, we consider such devices to be an ideal platform for supporting engineers with informal sketching and modeling tools which also allow them to stepwise refine and formalize their sketches.

In this paper we present a tool-supported approach that provides users with free-form sketching capabilities and allows them to annotate their sketches, thereby defining their notations and enabling a semi-automatic formalization of the sketches. Furthermore, we elaborate on the results of two experiments that focused on qualitative feedback regarding the usability and utility of our approach.

The remainder of this paper is structured as follows: We present our research goal and approach in Section 2. The tool prototype is described in Section 3. The two experiments and results are discussed in Sections 4 - 7. Section 8 presents related work, and Section 9 presents conclusions and future work.

2 Flexible Sketch-Based Modeling

2.1 Main Goal

The overall goal of our work is to unite the flexibility of unconstrained sketching with the power of formal modeling. In this context, we want to provide a tool-supported approach that (i) allows users to sketch any informal models, (ii) provides means for assigning syntax and semantics to sketched elements on the fly, and (iii) supports the semi-automated transformation of sketches into classic semi-formal models (e.g. a class diagram or a statechart) [4]. We envision our approach to allow for free, flexible sketching (as pen and paper does), and at the same time support a step-wise beautification and formalization of the sketches, thus avoiding the media break between early software engineering sketches and semi-formal models [5].

2.2 Key Requirements

Discussions with experts and the study of related work led to the following high-level requirements that we consider to be relevant for building a solution that fulfills end-user needs:

High flexibility: Users shall be able to sketch any kinds of diagrams. There should be no restrictions limiting users' expressiveness.

Natural sketching: The tool shall allow the use of input devices that give users a natural feeling for sketching, for example, a tablet PC with a pen or an electronic whiteboard.

Formalization capabilities: The tool shall support the transformation of diagram sketches into classic semi-formal models by, e.g. a semi-automated method, thus avoiding media breaks.

Speed: The tool shall allow fast creation and annotation of sketches. If the process is not faster than traditional sketching followed by model re-creation in a modeling tool, nobody would be motivated to use it.

The biggest challenge regarding our work is the aim to give users maximum flexibility in what they are sketching, not restricting them to any specific language or notation. Yet, a tool needs to "understand" what a user is sketching in order to perform any transformation of sketches into semi-formal models. We address this challenge by giving users the freedom to draw anything they want, and request that users assign meaning to sketches in due course with the help of annotations. These annotations will then be turned into corresponding metamodel elements.

2.3 Our Approach

We aim at a process that lets users switch arbitrarily between three work modes (Fig. 1): in the *modeling* mode, the user creates, augments or modifies sketches. In the *metamodeling* mode, the user annotates sketches and the tool creates metamodel elements based on these annotations. In the *sketch recognition* mode, the tool transforms sketched elements into semi-formal model elements by recognizing and interpreting sketches based on the information currently available in the metamodel. To make this process work, annotating must be easy and straightforward. No programming, scripting, or metamodeling skills should be required for this task. This free interleaving of modeling, metamodeling and recognition activities is the key difference between the presented approach and related work, where any form of metamodeling has to be done first. The following paragraphs depict our vision in more detail.

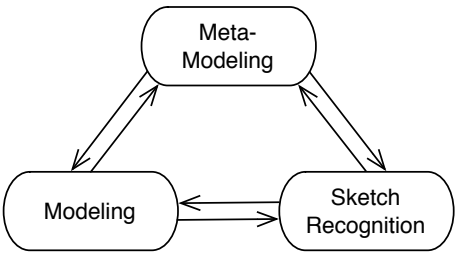


Fig. 1. The three activities leading to a semi-formal model in the end

Modeling is facilitated by two drawing modes. One mode mimics a whiteboard and allows for free sketching, while the other mode enables users to modify individual symbols (e.g. scale, move, copy). As opposed to other work, we do not define the modes as exclusive modes, i.e. the modes are not switched by pressing a dedicated button. The modes are switched implicitly. We believe that a single mode stays closer to the pen and paper metaphor. As soon as a user selects an already sketched object, she can make modifications.

Metamodeling starts when users assign a type to a previously sketched symbol. The symbol then gets added automatically to a dynamic symbol library. A symbol library consists of all defined symbols including their types, and allows to decouple types and their graphical representations. Symbol libraries are the first part of a lightweight, end-user metamodeling approach that supports users in defining a modeling language. Each symbol library contains a set of symbols belonging to a specific context, i.e. a specific diagram type. Users can define multiple contexts where the same symbol has different meanings. Because people like to mix different visual conventions during creative tasks [6], multiple symbol libraries can be active at the same time. A big challenge is to invent an easy-to-use interface for more complex metalanguage definitions, such as associations and rules for associations.

Sketch recognition happens in an interactive manner. Users are encouraged to take part in the recognition process to train the recognizer. They are also able to decide when and if recognition feedback should be given or not. These options assure that users do not get distracted from their sketching.

Our flexible process produces two kind of re-usable artifacts: the models drawn by the user, and a partial metamodel that is contained in the created (or re-used) symbol libraries. We envision that these two artifacts will make it easy to share models between different persons. Moreover, exporting the sketched models and further processing them in more formal SE tools comes within reach.

3 The FlexiSketch Prototype

We developed the FlexiSketch prototype, a mobile tool for freeform sketching with the focus on diagram sketching for SE. The prototype serves as proof of concept that the idea of letting the user switch flexible between all three activities works. In this section we present our prototype in detail, and we show how it realizes each of the three activities depicted in Figure 1.

Our tool is written in Java using the Android SDK for the Android OS, supporting version 3.0 and above[1]. We especially focused on tablet computers (with screen sizes around 10 inches) since we consider smart phone screens to be too small for effective sketching. Figure 2 shows a screenshot of our prototype.

[1] FlexiSketch is available on Google Play (formerly known as Android Market). A video demonstration can also be found on Google Play or at `http://www.youtube.com/watch?v=D06t0K5Otzw`

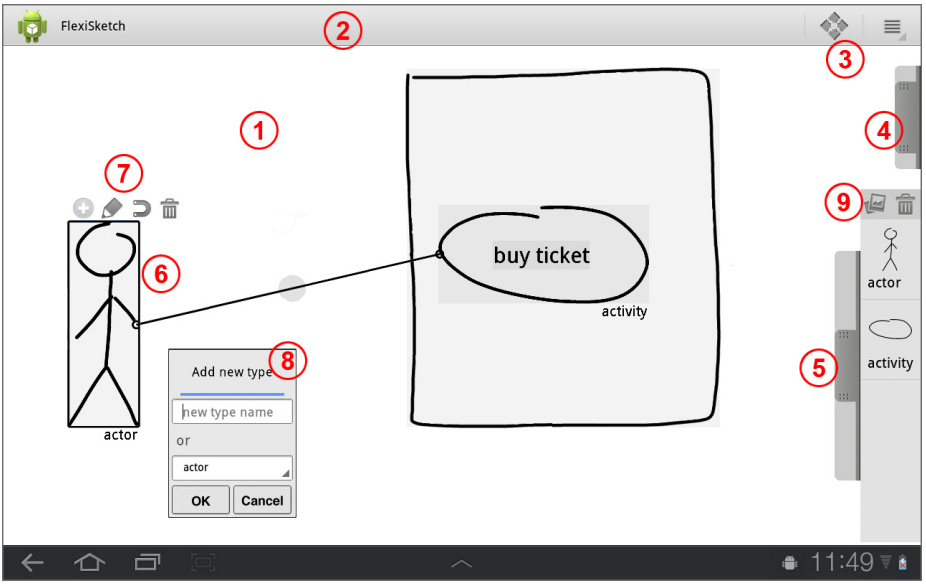


Fig. 2. Screenshot of the FlexiSketch prototype

3.1 Modeling

In order to emphasize the possibility of freeform sketching, the tool shows a white drawing canvas (1) when it is started. Additional functionality is hidden in the action bar at the top (2) and pull-out containers (4,5) on the right edge of the screen. To not distract the user from freeform sketching, we wanted to hide as much of the GUI as possible. However, we included a permanent action menu bar at the top of the screen to make some functions like scrolling quickly accessible. Scrolling of the drawing canvas is activated at the push of a button (3), and allows the user to draw sketches in the size of an A4 paper, independent of the actual screen size of the mobile device.

The user can start to draw like she would in any other paint application, either with her fingers or with a stylus. When the user stops to draw a particular item and removes the fingers from the screen for a predefined amount of time, the drawing is converted into a distinct symbol. The conversion of drawings into a symbol is not visible for the user per se, but the tool colors the background of a symbol with a light gray to give feedback that the conversion took place.

Symbols can be selected by tapping on them (touching the screen and lifting the finger without moving it). A selected symbol (6) is highlighted with a rectangular border around it and some context menu icons on top of it (7). The symbol can then be manipulated, dragged around, or deleted. The tool does not provide two distinct modes for drawing and symbol manipulation in order to allow for freeform drawing at any time. If the user touches the screen outside of the selected symbol and moves the finger, i.e. on white space or another symbol, she can just continue to draw naturally.

When the user starts sketching on top of one symbol and ends the stroke on top of another symbol, the tool recognizes the drawing as an association between the two symbols. It replaces the drawing by a straight line. In the middle of the line there is an anchor point that allows to select the line (otherwise the line would be hard to grab) and show its context menu. The line replacement is optional and can be switched off in the options menu.

An arbitrary amount of text boxes can be attached to symbols via the context menu (7). The text boxes are tied to the symbols, but can also be moved individually.

3.2 Metamodeling

The prototype provides means for defining symbols by assigning types to them. This is the basic functionality needed for the symbol library to work. After sketching a symbol, the user can choose to define its type via the context menu (7). A small popup menu provides the user with a list of all types that she already created, and an option to add a new type (8).

When the user decides to define a new type, she enters a name for it with the standard virtual keyboard. The tool creates a new entry in the symbol library with the given name, and stores the selected symbol in it. The symbol library contains entries for all defined types, and each type is graphically represented by at least one symbol (the symbol that was selected when the user defined the type). A list consisting of these graphical representations is shown in the pull-out container at the right edge of the screen (5). From there, defined symbols can be re-used with a drag-and-drop gesture.

The symbol library allows to add multiple symbols to the same type. This flexibility is needed because different users might draw different symbols to depict the same meaning. The opposite can also happen: the same symbol might have different meanings in different contexts, i.e. in different diagram types. Therefore, a symbol library is meant to be a collection of symbols belonging to a single diagram type. Symbol libraries can be stored and loaded (9) independently from the user sketches.

Theoretically, associations can have different types (similar to symbols), but in the current prototype, the number of association types is limited and bound to predefined line styles. The look can be quickly changed via a drop-down menu.

To help the user to not forget defining the symbols, the main menu (2) includes an option to visually highlight all symbols on the screen that do not yet have a type assigned.

Types assigned to symbols can be shown or hidden through the options menu. This allows the user to show types while she is taking care of type definitions. She can hide the types when she is working with text boxes, which do not belong to type definitions, but to particular instances of types. For example, in a class diagram most symbols are of the type class, but each symbol on the screen has a different class name written in a text box. To reflect the distinction between symbol types and text content of symbols, the functionality is strictly separated, accessible through two different context menu icons.

3.3 Sketch Recognition

We distinguish between *object recognition* and *sketch recognition*. We define object recognition as the automatic process of dividing user drawings into distinct symbols (and associations). The prototype does this while the user draws by using a simple timeout mechanism.

We define sketch recognition as the automatic process of comparing symbols to each other in order to identify similar symbols. For example, the user draws a stickman, and assigns the type *actor*. The sketch recognizer detects when the user draws a second stickman, so that the tool can automatically assign the type *actor* to it.

The prototype performs sketch recognition on drawn symbols. After the user draws a symbol, the sketch recognition algorithm searches the symbol library whether there are similar, already defined symbols. If the algorithm does not find any similar symbol, nothing happens. If a similar symbol is found, a pop-up on the bottom of the screen presents the symbol type for several seconds. If multiple similar symbols are found, the three closest matches are shown. The user can tap on one of the proposed types to assign this type to the newly drawn symbol. Choosing one of the proposed types is optional. The user can also decide to ignore the proposals and continue sketching. In this case the proposals will fade out again. This mechanic allows to give non-disruptive sketch recognition feedback to the user. If the user chooses one of the proposals, the new symbol is added to the respective type entry in the symbol library. In this way we realize a trainable sketch recognizer that learns from user inputs. The training is transparent to the user in order not to distract her from her actual task.

When the user draws a symbol that looks similar to an already defined symbol, an optional feature allows to replace the newly drawn symbol by the defined one. This can help to get a more uniformly looking sketch at the end. The feature can also be used to beautify symbols: A user can place one of the provided standard geometric shapes in the drawing canvas and assign a type to it. When she then draws a similar symbol by hand, the sketch recognition mechanism proposes to replace it with the geometric shape. Figure 3 shows a use case diagram where sketch recognition was used to beautify the hand drawn symbols.

For the implementation of the sketch recognition algorithm, we adapted an algorithm that is based on the Levenshtein string distance [7], which calculates the distance between two strings.

4 Evaluation

We used the developed prototype to evaluate our approach in early 2012. We performed two experiments to assess the usability and utility of our tool-supported approach. The perceived utility, or usefulness [8], affects the intention to use the approach, and therefore influences how much the approach will be adopted in practice [9].

We especially wanted to know whether our tool provides the same kind of flexible, ad-hoc sketching support that is also provided by paper and pencil-based

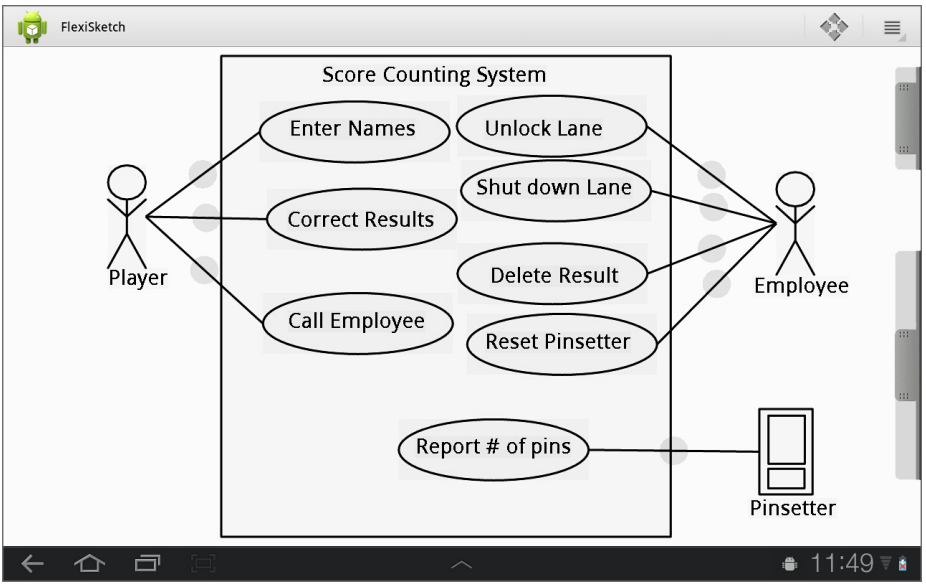


Fig. 3. Sketch recognition beautified a hand drawn use case diagram

approaches. Therefore we asked: What are the differences between FlexiSketch and paper and pencil-based approaches (*RQ 1*)? We also wanted to know whether users can in general classify the elements they draw with our tool into types (*RQ 2*). For adoption in practice, a tool has to be tailored to the needs of its users. Therefore we asked: Can they sketch diagrams with our prototype well enough such that they would consider adopting our tool in practice (*RQ 3*)? This also included to investigate what kind of diagrams software engineers sketch in practice.

The goal of the first experiment was to identify usability issues and assess whether our tool and approach allow users to sketch diagrams. In this experiment, we focused on answering RQ 1 and RQ 2 for one particular diagram type: use case diagrams. We chose this diagram type because (i) use case diagrams are widely known (and thus can serve as a common basis for our evaluation), (ii) they are typically used in early SE phases, and (iii) they are well-suited for transforming sketches into semi-formal models. This experiment did not allow to answer RQ 3.

The goal of the second experiment was to generalize from use case diagrams in order to answer RQ 1-3 for arbitrary diagrams. We wanted to investigate how the tool prototype can handle different diagram types (RQ 3). Another goal of the second experiment was to gain insights into how physical media for sketching are used in SE practice, and what kind of sketches practitioners draw.

In both experiments, participants used an Android tablet and a stylus to draw diagrams with our prototype and assign types to sketched symbols.

5 Experiment #1: Feasibility of our Approach and Tool Usability

This first experiment investigated the usability of our tool and the feasibility of our approach. The experiment was conducted in a controlled setting with 17 participants from research and industry: four undergraduate students and four PhD students in Computer Science, and nine software engineering practitioners, all having several years of experience (three persons counted as practitioners have returned from practice to academia). Practitioners were between 25 and 70 years old, and included experts in SE and HCI. 15 participants considered themselves to be novice users regarding touch interfaces (using smart phones for standard tasks, having no or little experience with tablet devices). Two persons considered themselves to be expert users (having a more in-depth understanding of mobile devices and their features, which includes tablet devices).

5.1 Method

The evaluation was conducted with each participant separately. It included a short briefing, the actual evaluation where participants had to perform three tasks using our tool, and a debriefing. In the briefing, we first asked demographic questions about the participant's knowledge in SE, touch interfaces, and working experience. Then we presented our research and gave a short introduction to the tool (about three minutes). For the first task, participants were asked to sketch a use case diagram according to a given problem description (our sample solution depicted five use cases). As second task, participants had to create a type for at least one instance of every distinct symbol As a last task, participants had to sketch a second use case diagram from a problem description of similar size, but this time using their predefined symbols whenever possible. For the tasks, we asked participants to think aloud. Interaction between us and the participants was reduced to a minimum, but participants were allowed to ask questions. We recorded our observations. In the debriefing, we asked about the usability and utility of the tool. Questions were based on the IBM Computer Usability Satisfaction Questionnaire [10].

The briefings lasted from five to ten minutes, working with the tool took between 20 and 30 minutes. Debriefings lasted from 30 minutes to one hour.

5.2 Results

Qualitative feedback from the participants and our observations revealed that they were able to draw use case diagrams with our tool. In debriefing meetings participants told us that they liked the informality provided by the tool. Furthermore, it took them a very short amount of time to learn how types are assigned to symbols and how the container with the symbol library is used (RQ 2).

When we asked about the symbol library that allows for the re-use of defined symbols, 14 out of 17 participants said that they like the feature and that they

perceive it to be more or equally efficient than sketching by hand (RQ 1). Three participants argued that use case diagrams consist of simple symbols easy enough to sketch by hand each time. But most participants said that even if the re-use mechanism is slower, they prefer it over sketching for the reason that all instances of the same symbol look exactly identical.

Almost all participants stated that it is still a bit faster to sketch with pen and paper, but the additional functionality of the tool compensates for it (RQ 1). Most frequently mentioned advantages over paper and pencil were: the manipulation of symbols, the re-use of symbols, the save/load option, and the (not yet implemented) possibility to export the sketches to other programs.

The following features were also liked and mentioned frequently: (1) the possibility to draw anything, (2) sketched lines between symbols turning into straight connections, (3) the use of different colors to distinguish symbols, (4) the option to merge two drawn symbols into a single one, and (5) the text autocompletion feature provided by the OS.

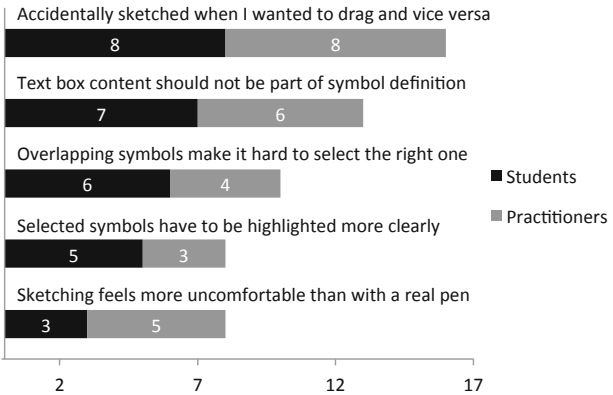


Fig. 4. The 5 most frequently occurred usability problems

In the following we discuss the most frequently mentioned usability issues (see also Fig. 4). Some of the issues are related to our design decisions and need re-thinking, while others are relatively easy to solve in future tool versions.

Sketching versus dragging: We do not provide two distinct modes for drawing and the manipulation of symbols. Participants often painted over existing symbols when they wanted to drag the symbols instead. Few said it actually makes more sense that the symbols have to be selected before they can be dragged around, but they continued to make the mistake.

Text boxes in symbol definitions: When users define symbols that contain text boxes, the text boxes - including the previously entered text - become part of the definition. This makes it possible to define, e.g. a class symbol with three text boxes. But participants said that it takes too much time to change the text

Table 1. The 8 most frequently mentioned feature wishes

	Students (8)	Prac. (9)	Total (17)
Larger screen (flip chart, whiteboard)	5	6	11
Symbol grouping for faster editing	5	5	10
Automatic beautification of symbols	5	5	10
Zoom / Scaling Function	3	7	10
Unlimited scrolling (larger canvas)	6	3	9
Resizing of drawn symbols	6	3	9
Landscape format	3	5	8
Ability to define types of associations	4	4	8

of the text boxes: copies of defined symbols should come without, or with empty text boxes.

Overlapping symbols: Multiple taps on overlapping symbols successively select each symbol in turn. But participants told us that they did not figure this out and therefore had problems selecting symbols overlapped by other symbols.

Selection highlighting: Participants reported that the visual cue for highlighted symbols was not strong enough. As implication, they accidentally manipulated the wrong symbol, especially when symbols overlapped.

Sketching does not feel natural enough: Indeed, participants mentioned that sketching does not feel the same as with paper and pencil. The tip of the used stylus is wider than a normal pen, and the tablet has a tiny, but noticeable lag before it displays the drawn strokes. This is due to hardware characteristics of the tablet.

There are several feature wishes that participants mentioned a couple of times and therefore got a high ranking (see Table 1).

Larger screen: The wish for a larger screen (in the size of a flip chart or whiteboard) made it to the top. Related to this are the following wishes (see Table 1): a *zooming or scaling function*, *unlimited scrolling*, *resizing of drawn symbols*, and the ability to turn the tablet to use it in *landscape format*.

Symbol Grouping functionality: More than 50% of the participants would like to have a grouping function, so that they can manipulate a group of symbols at the same time (e.g. to move them around or to delete them).

Beautification: 10 out of 17 participants wish to have an advanced beautification function. In particular, they were suggesting to have a feature that beautifies symbols based on their geometry. For example, a hand drawn rectangle should be displayed with straight lines, and a hand drawn oval should get smooth curves.

Different types of associations: Participants also stated the missing ability to define different types of lines, i.e. associations.

Further feature wishes include an undo feature, standalone text boxes and a distinct feature to add commentary text to the sketches.

5.3 Findings

With this experiment, we investigated whether the FlexiSketch tool can be used to draw use case diagrams, and whether it supports ad-hoc and flexible sketching.

Furthermore, we investigated whether users are able to define the individual elements of a use case diagram by assigning types to symbols (RQ 2).

All participants confirmed that they could draw the diagrams corresponding to the given problem description. They liked how the tool allows for freeform sketching. While observing the participants during the modeling task, we noticed three problems that delayed the completion of the task. These issues were also mentioned by participants.

First, in several cases participants accidentally drew a stroke when they wanted to drag a symbol. We will have to investigate whether this problem disappears once selected symbols get highlighted more clearly (which was also a mentioned critique). Therefore, we recently implemented a blue-colored background for selected symbols. We believe that users will associate the color with draggable symbols and will stop trying to drag symbols that are not colored blue. This is an assumption that yet needs to be evaluated. If the assumption does not hold, we have to rethink our design decision of not having two alternate modes for drawing and the manipulation of symbols.

Second, the screen size of the tablet computer proved to be limiting. Many participants lost time due to rearranging symbols, and they asked for a larger screen or an enhanced scrolling function. This seemed to be the biggest issue regarding adoption in practice.

Despite these issues, participants stated that drawing is easier and faster than with any other software modeling tool they know. Furthermore, we found the resulting diagrams to be well readable.

Participant feedback showed that they had no problem with defining the individual symbols (RQ 2). Some of them even started to define the symbols right away although we only asked them to draw a use case diagram as the first task. Around 50% of the participants, mainly students, asked how to make copies of symbols. Defining them is the only way in the current prototype. This might have been an additional motivation for the participants to do so.

Participants frequently mentioned the lack of certain tool features being a disadvantage, although physical media like paper and pencil do not have these features either (RQ 1). It seems that for paper and pencil, most people unconsciously accept the lack of symbol manipulation features. But they expect them to be present in a software tool. Participants stated that for them it felt a bit slower to draw with a software tool, thus a tool must compensate this by providing additional features. They mentioned that, once the formalization or export function is available, the whole process will be faster with our tool compared to paper and pencil.

6 Experiment #2: Utility of the Approach

This experiment gathered data about why and what kinds of diagrams practitioners sketch in their daily work, and assessed the utility of our approach. The experiment was conducted in a controlled setting with the nine practitioners from the first experiment. Thus, we refer to Section 5 for demographic information about the participants.

6.1 Method

The experiment was done with each participant separately, and included three parts: an interview, the evaluation and a debriefing. We started with questions about (1) how participants use sketches in SE practice, (2) what kind of diagrams they sketch, and (3) how these sketches are re-used later on.

The participants were then asked to sketch an example diagram of a diagram type they use frequently in their daily work, and to think aloud while using the prototype. Like in the first experiment, communication was reduced to a minimum, and we recorded our observations. After the participants completed their sketches, they had to define one instance of each occurring symbol. We stored the resulting symbol library and asked the participants to sketch another diagram of the same kind, re-using the previously defined symbols whenever possible.

At the end we asked debriefing questions about the usability and utility of the prototype, and how participants would like to further use and process the drawn sketches.

The interview, the briefing, and the debriefing together lasted for 30 to 40 minutes. Working with the tool took between 20 and 30 minutes.

6.2 Results

Qualitative feedback from the participants was very encouraging. All but one practitioner told us that they know situations where a polished version of the tool would be useful to them in practice (RQ 3). Two participants explicitly asked us to keep them up to date regarding our research and provide a more advanced tool version.

When we asked participants whether they use paper and pencil, and whiteboards in their work, they all answered yes. Whiteboards and flip charts were mentioned to be more important than paper and pencil.

We also wanted to know from participants what kinds of diagrams they draw in practice on whiteboards and paper (RQ 3). Table 2 summarizes the answers. Participants answered that they do not just use certain diagram types, but they draw *"something similar to, but not quite a..."* or *"a simplified version of a..."* followed by the diagram type. Regarding UML models, use case diagrams were mentioned because they are *"something that stakeholders understand"*. Many participants said they draw diagrams to show some kinds of processes, e.g. business processes, and to show structures and relationships or transactions between stakeholders and systems, or between systems and subsystems. Two participants emphasized that they draw different kinds of diagrams dependent on their stakeholders' knowledge.

When we asked for the reason of using whiteboards or paper, the most frequent answer was that the large size of whiteboards and flip charts facilitates communication and reaching consensus (to reduce misunderstandings) in meetings and workshops (see Fig. 5). Participants also found it important to mention that whiteboards allow for collaboration and encourage attendees to participate.

Table 2. Diagrams drawn on whiteboards and paper as reported by participants

- Process models (block diagram with events and activities, BPM)
- Business models (clients, orders, storage dimensions)
- Mindmaps
- Flowcharts
- Architecture diagrams (layer cake)
- Use Case Diagrams
- System architecture models (dependencies between modules)
- Boxes and arrows
- Simplified activity diagrams (Boxes connected with signals)
- Transactions (systems and flows of information)
- Entity diagrams
- No particular diagram type, dependend on stakeholders
- Sequence diagrams

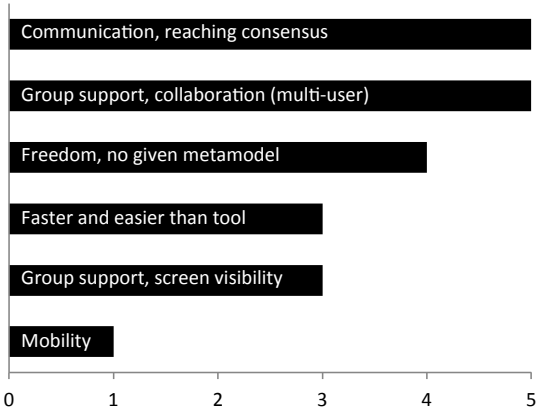


Fig. 5. Most frequently stated reasons why paper or whiteboards are used

The next reason for using physical media is that it supports freeform sketching. Participants mentioned that they find it faster and easier to draw on physical media compared to software tools. They stated that a large screen is preferred because many people can look at it at the same time.

We asked participants how the sketches drawn on physical media are re-used later on. Four participants said that they take a photograph and later re-create a model in another software tool using the photo as reference. Three participants stated that the content from the sketches is either communicated verbal, or a certain kind of documentation is created. Two participants usually take a photograph and put the picture directly into another document, where they add descriptions to it. Also, two participants told that in many cases the sketches only remain valid for some weeks, and are not used anymore afterwards (as they cannot be edited).

After the interview, participants used our tool to draw simple examples of their most used diagram types (RQ 3). Feedback about the usability and utility was similar to the one from the first experiment. *"A larger screen is needed"* was one of the most frequently given answers when we asked about their willingness to use an improved version of the tool in practice. Apart from this and the already mentioned usability issues, they were positive about a possible adoption in practice.

After participants have drawn diagram examples, we asked them what features exactly they would expect from an export function. Here, the answers were quite diverse: some just want to be able to distribute it. Some want to have beautified versions of the sketches, while others are happy with the sketchy look. Most of the participants want to have it in an editable form (either beautified or not). One participant would like to export a list of the sketched entities into MS Excel. Another participant wants to work iteratively on the sketches, having the option to synchronize them back an forth between our tool and another software tool.

6.3 Findings

With this experiment, we investigated what practitioners usually sketch in practice, whether these sketches can be drawn with the FlexiSketch tool, and whether practitioners like our approach, i.e. would be willing to adopt our approach in practice. We also wanted to see whether they can classify the symbols they draw into different types.

We conclude that participants liked our approach as it suits the kinds of diagrams they draw (RQ 3): no standard diagrams, but something similar. Participants also frequently said that they draw UML-like diagrams, introducing their own notation.

Participants were also able to categorize the drawn symbols by assigning types (RQ 2). We were afraid that users might not be able to handle even this first step of lightweight metamodeling, but some participants actually wanted to do even more. For example, they asked how they could define different types of associations, constraints, and how to declare a type as subtype of another one. However, other participants did not bother about metamodeling. This highlights the diversity of user skills which our approach should account for.

We probably have to trade the mobility of tablet computers against a larger screen (e.g. an electronic whiteboard) to gain a wider adoption in practice. Electronic whiteboards allow for multi-user input and thus facilitate collaboration, but they are far less wide-spread than mobile devices. Results suggest that the ability of a sketching tool to facilitate communication is important. This corresponds to findings from Ossher et al. [3].

When it comes to the question whether we should allow for an easier formalization of sketches by limiting the freeform sketching, participants agree that the freeform drawing is more important than formalization capabilities.

7 Threats to Validity

Construct Validity. We concluded that users are able to assign types to symbols. Depending on how we intend to transform informal sketches into semi-formal models, it might not be valid to state that users are able to provide relevant metadata if they are able to assign types. So we might be measuring what we mean to measure in special cases only.

Internal Validity. Participants voluntarily assigned types to symbols, as it was the only way to make copies of a symbol. This additional benefit can be seen as a motivating factor to define symbols, and therefore is a threat to internal validity.

Conclusion Validity. We did not try to measure a relationship between some treatment and some outcome, nor did we try to calculate statistical significance in our data. We were rather interested in assessing the feasibility of our approach and the utility of our tool.

External Validity. Our goal was to gather qualitative rather than quantitative data. However, the small sample size of our experiments is a threat to external validity. Many of the practitioners in our experiments have an academic background, and might therefore be more enthusiastic about our approach than software engineers in general. In the second experiment, participants decided to draw rather simple and high-level sketches because of time constraints and screen size limitations. Although these sketches contained key elements of drawings they use in real-world projects, this can be seen as a threat to external validity.

8 Related Work

Various mobile software tools for freeform sketching can be found online (e.g. Developer Whiteboard [11]). There are also modeling tools for both general-purpose and SE-specific modeling. For example, DroidDia [12] allows to create different diagrams like flow charts, Venn diagrams, mind maps, and so on. It provides predefined symbols as well as basic geometric shapes. Although mobile computing is a recent trend in SE (e.g. [13]), we are not aware of any existing work about mobile software tools that let users define and annotate their own diagram symbols. We fill this niche with our FlexiSketch approach.

Looking at desktop tools for sketch-based modeling support in SE, there are two threads of related approaches: (1) augmenting formal modeling tools with sketch recognition features, and (2) augmenting informal modeling tools (e.g. a tool mimicking a whiteboard) with features that allow a certain degree of formalization.

In thread 1, various approaches and tool prototypes have been developed that allow users to sketch diagrams (e.g. [14,15]). The key idea is that sketch recognition algorithms will convert the produced sketches into semi-formal models. However, this also means that a user, when drawing a sketch, is still following the tool's predefined notations. Users therefore have to understand a specific

modeling language in order to produce sketches that can be converted automatically. The sketch recognizer cannot interpret sketches that do not adhere to the language. Therefore, these approaches limit creativity and expressiveness. They also distract users from the actual modeling task [16] and can cause additional overhead due to sketch recognition errors.

Approaches following thread 2 seem to be more promising. However, there are several challenges. For example, the set of possible sketches increases as soon as the user gains freedom in sketching. Thus, it gets more difficult for a tool to analyze and identify what the sketches actually mean. So far, only few researchers have tackled these issues and built tool prototypes that support users in drawing sketches independently of a specific modeling language. Such an example is the Calico prototype [17]. It provides mechanisms to structure sketches into different parts. Furthermore, a user can connect the parts with arrows. However, the user is not able to assign some meaning to the sketched symbols.

Other tools such as MetaEdit+ [18] and MaramaSketch [19] include metamodeling editors. These editors allow to define a custom modeling language. The language definition is then used to compile a modeling tool for the defined language. However, these tools require to create the full language definition first and then users must strictly adhere to it, thus preventing any flexible sketching.

Some approaches do not require users to define symbols at the beginning. For example, BITKit [3] is a flexible modeling tool that focuses on combining the advantages of office and modeling tools. However, it does not include freeform sketching. The Electronic Cocktail Napkin [1] is a freehand drawing environment for conceptual design, and allows to incrementally transform sketches into schematic drawings. It does not focus on SE, but on architectural design (e.g. buildings). The open source project Sketch for Eclipse [20], currently under development, is an API that provides sketching capabilities, a trainable gesture recognizer, and allows users to classify sketched elements. However, the project does not focus on user metamodeling. Since it is meant to be an API, it is not suited for end-users.

In summary, there is no existing solution that allows ad-hoc modeling and satisfactorily bridges the gap between freeform sketches and semi-formal models.

9 Conclusion and Future Work

In this paper we report on a tool-supported solution that combines freeform sketching with the ability to interactively annotate the sketches for an incremental transformation into semi-formal models. Software engineers using our approach have the possibility to evolve sketches into semi-formal models rather than re-modeling the information contained in the sketches in a software modeling tool. Our experiments provide first answers to our research questions: RQ 1: What are the differences between our tool and paper and pencil? User feedback indicated that sketching with FlexiSketch "feels" less natural and slower compared to paper and pencil, but additional functionality provided outweighs these limitations. RQ 2: Can users in general classify the elements they draw

into types? Participants in our experiments had no problems doing so, therefore we answer this question with yes. RQ 3: Would users consider adopting our tool in practice? All but two participants were positive about adopting the tool in practice. We cannot generalize this result for a larger user-base, so we answer with a tentative yes.

The major contributions of this paper are:

- A new, flexible process for sketching and the step-wise formalization of these sketches. The core of our approach is not specific to the SE domain, but can also be valuable for many other domains.
- A tool prototype that highlights the feasibility of our approach. Users can sketch first and assign meanings to the sketches on demand.
- Two initial usability and utility experiments showing that software engineers are able to sketch their favorite diagrams with our tool, and can annotate them.
- Insights about the use of informal sketching approaches in SE practice.

Recent technological trends and the growing market of tablet PCs provide the opportunity to come up with novel solutions supporting flexible, mobile, ad-hoc modeling in SE. Our work describes one possible approach how tool support for early SE phases may look like.

Although first evaluations indicate the usefulness of our approach, more research is needed to gain answers to open questions. Our future research will focus on the following:

Hardware for Sketching. Modern tablet computers usually come with a capacitive touch screen. Digital pens must have wide tips in order to get properly recognized by the tablets. For some users, sketching with these pens feels unnatural. We need to investigate how we can make sketching feel more natural. Moreover, the small screen sizes of tables are an issue. Electronic whiteboards provide a large drawing space and facilitate collaboration, but limit end-user mobility and are far less pervasive than mobile devices. At the moment, a native Java version of FlexiSketch that runs on desktop machines and electronic whiteboards is planned, but our main focus remains on supporting SE activities with mobile devices.

End-user Lightweight Metamodeling. It is an open challenge how metamodeling can be made accessible to end-users [3]. Assigning types to symbols is only a first step towards a lightweight, end-user metamodeling approach. It is thus worthwhile to investigate how a more complete end-user lightweight metamodeling method could look like, and how it can be integrated into our approach. We want to explore how much information relevant for metamodeling is put into model sketches by users themselves, how much metamodeling is needed for exporting models into other software tools, and whether the tool could infer the missing information (semi-)automatically.

Field Studies. So far, we performed interviews and two controlled experiments. Once an improved version of our tool is ready for use in practice, we have to conduct case studies where our tool is used in the field during real-world projects. This allows to further assess the utility and adoption in practice of our approach.

We want to explore in which situations our tool can be used, and how it will be used. The studies will point out the benefits and limitations of our approach in more detail, and will enable us to refine it.

Acknowledgment. The authors would like to thank Sebastian Golaszewski, who made a big contribution to the FlexiSketch tool.

References

1. Gross, M.D., Do, E.Y.-L.: Ambiguous intentions: a paper-like interface for creative design. In: Proc. 9th ACM Symp. on User Interface Software and Technology, pp. 183–192 (1996)
2. Branham, S., Golovchinsky, G., Carter, S., Biehl, J.T.: Let's go from the whiteboard: supporting transitions in work through whiteboard capture and reuse. In: Proc. 28th Int. Conf. on Human Factors in Computing Systems, pp. 75–84. ACM, New York (2010)
3. Ossher, H., Bellamy, R., Simmonds, I., Amid, D., Anaby-Tavor, A., Callery, M., Desmond, M., de Vries, J., Fisher, A., Krasikov, S.: Flexible modeling tools for pre-requirements analysis: conceptual architecture and research challenges. In: Proc. ACM Int. Conf. on Object Oriented Programming Systems Languages and Applications, pp. 848–864 (2010)
4. Wüest, D., Glinz, M.: Flexible Sketch-Based Requirements Modeling. In: Berry, D., Franch, X. (eds.) REFSQ 2011. LNCS, vol. 6606, pp. 100–105. Springer, Heidelberg (2011)
5. Wüest, D.: Bridging the gap between requirements sketches and semi-formal models. In: Doctoral Symposium of the 19th IEEE Int. RE Conf. (2011), http://dx.doi.org/10.5167/uzh-55675
6. Cherubini, M., Venolia, G., DeLine, R., Ko, A.J.: Let's go to the whiteboard: how and why software developers use drawings. In: Proc. SIGCHI Conf. on Human Factors in Computing Systems, pp. 557–566 (2007)
7. Coyette, A., Schimke, S., Vanderdonckt, J., Vielhauer, C.: Trainable Sketch Recognizer for Graphical User Interface Design. In: Baranauskas, C., Abascal, J., Barbosa, S.D.J. (eds.) INTERACT 2007. LNCS, vol. 4662, pp. 124–135. Springer, Heidelberg (2007), http://dl.acm.org/citation.cfm?id=1776994.1777013
8. Moody, D.L.: The method evaluation model: a theoretical model for validating information systems design methods. In: Proc. 11th European Conf. on Information Systems, pp. 1327–1336 (2003)
9. Moody, D.L., Sindre, G., Brasethvik, T., Sølvberg, A.: Evaluating the quality of information models: empirical testing of a conceptual model quality framework. In: Proc. 25th Int. Conf. on SE, pp. 295–305 (2003)
10. Lewis, J.R.: IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. Int. J. Hum.-Comput. Interact. 7, 57–78 (1995)
11. Developer Whiteboard, https://play.google.com/store/apps/details?id=com.agilaz.whiteboard
12. DroidDia, http://www.droiddia.com/
13. Seyff, N., Graf, F., Maiden, N.: Using mobile RE tools to give end-users their own voice. In: Proc. 18th IEEE Int. RE Conf., pp. 37–46 (2010)
14. Chen, Q., Grundy, J., Hosking, J.: SUMLOW: early design-stage sketching of UML diagrams on an e-whiteboard. Softw. Pract. Exper. 38(9), 961–994 (2008)

15. Hammond, T., Davis, R.: Tahuti: a geometrical sketch recognition system for UML class diagrams. In: AAAI Spring Symposium on Sketch Understanding, pp. 59–68 (2002)
16. Alvarado, C., Davis, R.: SketchREAD: a multi-domain sketch recognition engine. In: Proc. 17th ACM Symp. on User Interface Software and Technology, pp. 23–32 (2004)
17. Mangano, N., Baker, A., Dempsey, M., Navarro, E., van der Hoek, A.: Software design sketching with Calico. In: Proc. IEEE/ACM Int. Conf. on Automated Software Engineering, pp. 23–32 (2010)
18. Kelly, S., Lyytinen, K., Rossi, M.: MetaEdit+: A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment. In: Constantopoulos, P., Mylopoulos, J., Vassiliou, Y. (eds.) CAiSE 1996. LNCS, vol. 1080, pp. 1–21. Springer, Heidelberg (1996)
19. Grundy, J., Hosking, J.: Supporting generic sketching-based input of diagrams in a domain-specific visual language meta-tool. In: Proc. 29th Int. Conf. on SE, pp. 282–291 (2007)
20. Sangiorgi, U.B., Barbosa, S.D.: Sketch: modeling using freehand drawing in Eclipse graphical editors. In: ICSE 2010 Workshop on Flexible Modeling Tools, Cape Town, South Africa (2010), http://www.ics.uci.edu/~tproenca/flexitools/papers/10.pdf

Achieving Targeted Mobile Advertisements While Respecting Privacy

Elia Palme[1,2], Basil Hess[2], and Juliana Sutanto[2]

[1] Newscron, Lugano, Switzerland
[2] ETH Zürich, Management Information Systems
Zürich, Switzerland
{epalme,bhess,jsutanto}@ethz.ch

Abstract. Broadcasted mobile advertisements are increasingly being replaced by targeted mobile advertisements through consumer profiling. However privacy is a growing concern among consumers who may eventually prevent the advertising companies from profiling them. This paper proposes an agent-based targeting algorithm that is able to guarantee full consumer privacy while achieving mobile targeted advertising. We implemented a grocery discount-discovery application for iPhone that makes use of the new approach. We show that on modern hardware like on the iPhone, it's feasible to run a client-based and privacy-preserving targeting algorithm with minimal additional computational overhead compared to a random advertising approach. We evaluated the targeting method by conducting a large-scale field-experiment with 903 participants. Results show that the computational overhead on user devices is well tolerated, compared to the control group with randomized advertising the targeting group showed a significantly increased application usage of 18%.

Keywords: mobile pervasive advertisement, privacy, targeted one-to-one marketing, mobile agent.

1 Introduction

Advertisements of the future are likely to be mobile and highly targeted [5]. Mobile advertising refers to advertising using mobile devices such as tablet pcs, smart phones or cellular phones [7]. The International Telecommunication Union (ITU) estimated 5.3 billion cellular subscribers at the end of 2010, including 940 million subscribers to 3G services. The high penetration of mobile devices enables merchants to reach a large number of potential customers. Gartner estimates 70 billion mobile app. downloads for 2014, compared to 17.7 billion in 2011[1].

Besides the term "mobile", another popular term in the advertising industry is "targeting". Researchers have consistently shown that targeted advertising messages are more effective than the non-targeted ones [1]. Furthermore targeted advertisements can minimize consumers' annoyance level, and mobile device is

[1] http://www.gartner.com/it/page.jsp?id=1826214

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 245–263, 2013.

a good platform to achieve this aim. A mobile device "follows" its owner anytime anywhere. Because of the awareness of its owner's context and preferences, mobile devices are a useful medium for implementing targeted marketing.

In this paper, we present a concrete solution to achieve targeted mobile-advertising while respecting the consumer need for privacy. Specifically, we propose an unsupervised, targeted mobile advertisement framework. We profile individual consumers and use their profile information to select the most captivating advertisement messages for each consumer while at the same time ensuring their privacy. In our framework, we develop a mobile advertising-agent that reaches to the customers and estimates their interests in the advertisement messages of their mobile device.

Customer privacy is protected through a distributed mobile framework in which a customer profile is locally stored in her mobile device, and the advertisement targeting is performed by her respective mobile device. The level of privacy protection that our method achieves is the following: Users first have to explicitly authorize local storage of their profile information. If authorized, we consider the internal storage of a mobile device to be trusted by its owner, so that local storage of private profile information on the phone doesn't affect privacy. The privacy-policy however guarantees that none of this information is ever transferred through the device's external communication facilities (e.g. WiFi, 3G) to any second party.

In the next sections, we review what other researchers have accomplished in this area, and introduce our mobile advertising-agent. This is followed by a detailed explanation of our proposed framework, and the results of our field experiment.

2 Related Work

While much work has been done on introducing the targeted advertising method to mobile devices, no study has so far proposed a technical solution to overcome the trade-off between targeted advertising and privacy concerns, and no large-scale field study on targeting methods has been done. Most of the studies on targeting are conceptual, presenting architecture design without details on how targeting is actually achieved. Our study on the other hand demonstrates that the flexibility provided by the mobile agent-architecture can be efficiently applied to the problem while transferring an acceptable computational overhead to the customer device.

We classify previous works based on three dimensions: 1) their advertising targeting method, i.e., how customer information is used to compute the matching between the advertisements and the customer, 2) their privacy handling, i.e., how the system protects customer privacy, and 3) how the system performance and effectiveness are tested.

2.1 Advertising Targeting Methods

Location filtering is the most basic method to select the most appropriate advertisements for the customers.

Location-Based Targeting. [2][6][25][9] uses one single piece of information to perform the targeting by comparing the customer's location with the ads target location.

Rule-Based Targeting. [3] will only display the advertisements if the customer's profile satisfies the ad's targeting rules, e.g., a gender rule.

Category-Based Targeting. [19][9][11][4][24][12] groups ads into categories. By analyzing customer profiles, it identifies which category is potentially interesting for the customer.

Ontology-Based Targeting. [16] defines hierarchical relationships among entities (user data and ad description) and tries to find the ad that has the highest number of related entities to the user profile.

Agent-Based Targeting. [14] is the most flexible method, because it allows to implement arbitrary targeting methods. Mobile agents carry interpretable code which allows them to implement any of the previously discussed strategies. For instance, an agent could simply check for the customer's location (Location-based targeting) or compute more elaborated targeting by identifying if the ad category fits with the customer profile. In our method an agent is used to compute an integer value that represents the degree to which an advertisement matches the customer's profile. More details on how mobile agents are used are in section 3.1. Mobile agents have the singularity of excuting the targeting code on the customer's devices; the evaluations will provide valuable insights on the agents computation overhead and the customer's acceptance of it.

2.2 Privacy Handling

To mitigate consumer's privacy concern, several distributed mechanisms have been proposed. These mechanisms try to reduce the consumer's concern either by providing high level of transparency on the collected data or by protecting the consumer's identity. We classified previous works on privacy handling in the following way:

Permission-Based Mechanism. [2] asks for consumer permission to collect specific information; it consequently provides high transparency on the amount and type of the collected data. In 1997 the World Wide Web Consortium (W3C) launched the platform for the Privacy Preferences Project (P3P). The main goal of P3P is to provide an automated mechanism to inform users on their private information treatment. Langheinrich [13] developed a Privacy Awareness System (PawS) based on P3P providing high level of awareness and "data collection" service opt-out option.

Table 1. Comparison of our study with related work on mobile advertising

	Targeting Method	**Privacy Handling**	**Validity Test**
Ranganathan and Campbell (2002) [19]	Category-based	Local Profiling	None
Aalto et al. (2004) [2]	Location-based	Permission-based	35 participants
Mahmoud et. al. (2007) [14]	Agent-based	None	None
Barnes et. al. (2008) [6]	Location-based	None	None
Gao and Ji [9]	Category-based, Location-based	None	None
Yang et. al. [25]	Location-based	None	137 students
Narayanaswami et. al. [16]	Ontology-based	Anonymity	None
Shannon et. al. (2009) [21]	Unknown	Aggregation	None
Aggarwal et. al. [3]	Rule-based	Local Profiling	None
Guha et. al. [11]	Category-based	Local Profiling	None
Alt et. al. [4]	Category-based	Anonymity	None
Toubiana et. al. (2010) [24]	Category-based	Local Profiling	None
Haddadi et. al. (2010/11) [12]	Category-based	Local Profiling	None
This paper	**Agent-based**	**Local Profiling**	**903 participants**

Aggregation Mechanism. [21] groups customers into clusters based on their demographic profile (e.g. a cluster of young people) and then targets the ad based on the cluster information. The privacy protection of this method is limited because it involves collecting consumer data.

Anonymity Mechanism. [16][4] protects consumer's privacy by anonymizing the data collected. However by collecting a large enough amount of anonymized data, the consumer's identity could be reverse engineered.

Local Profiling Privacy Enhancement. [19][3] [11][24][12] mechanism protects customer identity by avoiding the transmission of any data. Usually this method is coupled with other security mechanism (e.g. sand-boxed environment, anonymous pings or delay tolerant networks [12]) to guarantee that no information can be stolen or recovered while being able to bill for the ads impressions. This study adopts the local profiling method.

2.3 System Performance and Effectiveness Test

Most of the previous studies are conceptual in nature without actual testing of their proposed methods. Aalto [2] and Yang [25] validated their studies with a small amount of around 100 participants. In contrast, we propose a system that has been fully implemented and validated through a field experiment

(with around 1000 participants) to test the performance and effectiveness of the proposed method. Especially we demonstrate that mobile agents are efficiently used to distribute the targeting computation on the customer's devices. Agents have a minor impact on the customer devices performance and are therefore tolerated by users. The following sections provide insights on our system architecture, implementation, and evaluation.

3 Privacy-Preserving Targeted Advertising Framework

In this section, we present our privacy-preserving targeted advertising framework that dispatches, targets, and distributes advertisements in a privacy-preserving way. Three components compose the framework: the mobile advertising-agent (Section 3.1), mobile advertising-agents' client (Section 3.2), and mobile advertising-agents' server (Section 3.3).

Our framework is based on the following main principles: (1) Privacy-enhanced mobile agency for targeting advertisements, (2) a central advertising-agents server, and (3) mobile devices with capabilities to process agents (adver-tising-agents client). The mobile agents are privacy-enhanced because they don't leak any user profile data to any second party. Each agent corresponds to one advertisement, plus the targeting mechanism associated to it, whose goal is to find matching customers. Finally the user's mobile device decides if the agent with its advertisement and targeting method matches the user's profile.

As the matching between the carried advertisement and the consumer profile is performed on the client's mobile device, the number of consumers that can be handled is highly scalable and the server's infrastructure is very light. The process of creating an agent and displaying the advertisement is illustrated in Figure 1.

3.1 Mobile Advertising-Agent

The mobile agent comprises two separate and distinct concepts, i.e., mobility and agency [18]. Mobility denotes the ability to move and visit multiple hosts. Agency is the characteristic of an agent, who is hired by a principal (in our case an advertiser) to sell its product to interested customers (in our case the mobile device user). Upon reaching a customer's mobile device, a mobile advertising-agent checks if the customer is in the target group of the carried advertisement.

The mobile advertising-agent is a small piece of binary data, which carries a digital advertisement and a match estimator, as shown in Figure 2. The match estimator is an interpretable code used to select only those people that are within the target of the carried advertisement. Principally the mobile agent concept [17] comprises the ability of agents to autonomously traverse multiple hosts. However, to mitigate security risks [8], we restrain this feature and allow the migration process only through the central server. This allows us to control the migration flow and to trigger security checks. Each agent has to be cryptographically signed by the principal of the agent (i.e. the advertiser). Before delivering an agent, the

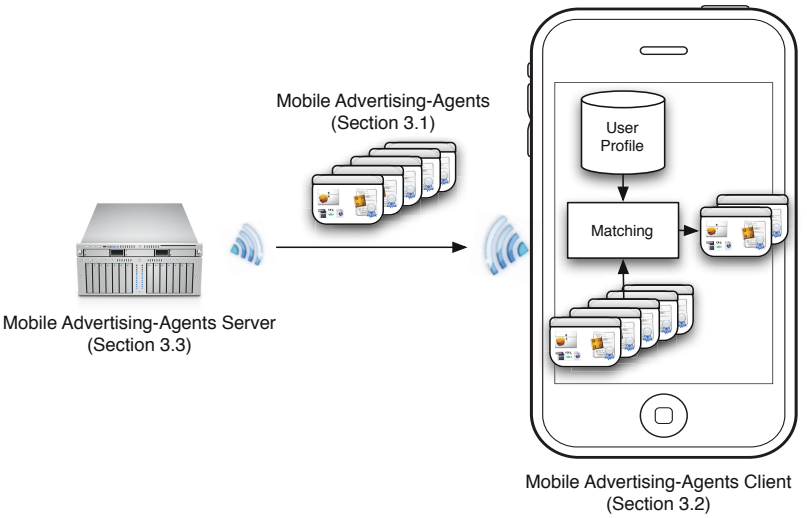


Fig. 1. The privacy-preserving targeted advertising framework, consisting of the server part (left), the mobile advertising-agents (middle), and the client part in a smartphone device (right)

Fig. 2. Schema of the Mobile Advertising-Agent. An agent comprises of a digital advertisement and a matching estimator code, which are cryptographically signed.

server verifies the cryptographic signature of the agent to be able to guarantee that only authorized advertisements are circulated. The server then singes the agent with its own signature, which is eventually verified by the client upon reception of the agent.

Match Estimator. The match estimator is a piece of software code that is carried by the agent. The code is originally generated by an advertiser, with the goal to determine if a user belongs to his target group or not. As part of the agent, the match estimator is transferred from the central server to the customer's

mobile device, where it is eventually executed. Match estimator can be any type of executable code. We prefer to use JavaScript [22] because many mobile platforms are offering APIs to run the scripts in a sandboxed environment. In our implementation iOS offers to run JavaScript in a sandboxed WebKit-instance that prevents all Internet connections, and protects the user profile from being modified.

The goal of match estimator is to determine the customer's affinity with the advertisement target. The outputs of the match estimator code are two integer values: *affinity coefficient* and *uncertainty value*. The higher the affinity coefficient, the closer the customer is to the advertisement's optimal target. The higher the uncertainty, the less accurate the computed affinity coefficient is. The uncertainty value depends on the availability and importance of the information needed to compute the affinity coefficient, which means that the value will be higher if the user didn't disclose personal data used by the match estimator. In order to compute the affinity coefficient, match estimator has to access the local information repository on the client side.

The local information repository is unique for each client and following our privacy policy, it is not sharable or accessible outside the client. In the information repository, personal information such as age, gender, willingness to buy budget products, or life style diet, is stored. Before interpreting the match, a repository is assembled at the client side. Assembling the repository means extracting personal information from whatever storage support is offered by the smartphone (e.g. in iPhone, the information can be stored and retrieved through the core data framework) and generate a dictionary that is readable by the match estimator code. The generated dictionary should be read only and accessible by the match estimator code as a global variable in the same sandbox environment where the match estimator is interpreted.

Finally the "evaluate" method of the match estimator code is called, the affinity coefficient and uncertainty value is computed and returned. The decision of displaying or dropping the advertisement is mainly based on the returned affinity coefficient. If the uncertainty value is too high, the client will postpone this decision and re-run the match estimator as soon as new information is available on the local repository.

We illustrate the principles of the match estimator with a short example: A beer producer wants to advertise a new sweet beer especially conceived for females. In order to make the advertising companying as much effective as possible the beer producer has an interest to only advertise its product to major age female customers. He therefore needs to purposely configure a mobile advertising agent by setting the correct match estimator code. The match estimator code takes the local customer profile (the repository where all private user information are stored) as input and returns the affinity and incertitude value as output. In this case the affinity is at maximum if the customer is a major age female. On the other side the affinity is at minimum if the customer is too young. It can happen that on the local repository some required information is missing, for example the customer's age in not present. In this case the match estimator

code cannot compute the affinity value or can only partially compute it. If all information required to compute the affinity are available the uncertainty value is zero. But in the case of the sweet beer if the customer age is missing the incertitude is at maximum because knowing if the customer is major age is a required information. If the uncertainty is too big the advertisement is put on hold till the required information is available. Besides this simple example, the interpretable code allows to achieve much more complex targeting.

Digital Advertisements. As illustrated in Figure 2, each mobile advertising-agent carries a digital advertisement. Modern advertisements rely on several types of media types like text, images or videos, and it should be possible to support all these cases. An important restriction is that no content must be loaded from remote servers, all contents have to be embedded in the advertisement itself. This implies two things: consumer privacy is protected, and the amount of transferred data is controlled. Privacy is protected because without explicit advertisement download, it is not possible to track who opens which advertisement. If all media contents are embedded in the mobile advertising-agent, we control the total data throughput to prevent consumers from downloading massive amounts of data. Finally, having all contents contained and stored in the digital advertisements allows displaying them even if Internet connectivity is lost at a later point in time.

In our implementation we choose to encode the advertisement as a HTML document to give us the liberty in defining the advertisement's graphical design. A digital advertisement is basically a webpage that can be composed by rich text and images. By choosing HTML to represent the advertisement, we "uniform" the encodings to some extent. Since our match estimators are JavaScript-based and the advertisement is HTML-encoded, we can manage both of them with the same web browser engine instance and reduce the client's memory consumption. When no external content is downloaded, external references in HTML would usually not be available. To overcome this limitation, we use an artifice provided by the data URI scheme to include in-line data in the HTML documents. The URI scheme is defined in RFC 2397 [15] and adopted by most browsers.

3.2 Mobile Advertising-Agents' Client

The mobile advertising-agents' client is the software part running on the smartphones (the client side of our framework architecture). The client is a place where: 1) mobile advertising-agents are hosted and their match estimator code is interpreted, 2) consumer's profile is stored, and 3) advertisement is displayed. The agents' client is integrated inside an application (e.g., a gaming or utility application) that would display the targeted advertisements.

Advertisements Selector. The client is responsible for fetching new mobile advertising-agents from the main server. All new agents are transmitted to the client that will then store them in the local database (see Figure 3). Newly transmitted advertising-agents are marked as unrated. All unrated agents' matching

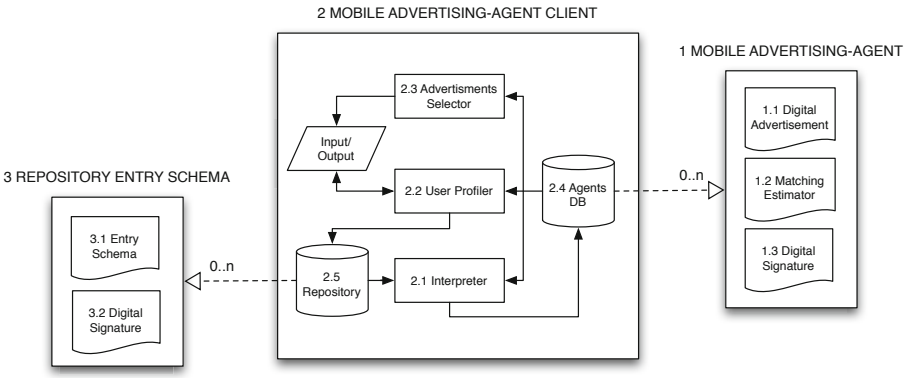


Fig. 3. A schematic overview of the Mobile Advertising-Agents' Client, showing the relationship between agents (1), client (2), and the client's local repository (3)

estimators are interpreted to establish their affinity coefficient and uncertainty value. After all match estimator codes of all new agents are computed, the client will proceed to an initial screening, i.e., only those agents that have a uncertainty value below a certain threshold will be considered. Those agents having an appropriate uncertainty value are than eligible for advertisement selection. The advertisement selection algorithm depends on the kind of advertising strategy the application wants to implement. For example, if the mobile advertising-agent client is embedded in a game application, it may show a small banner for a certain period of time. In this case, the advertisement selection algorithm could be: Take the advertisement with the highest affinity coefficient and display it for a certain amount of time, then take the advertisement with the second highest affinity coefficient, and so on.

User Profiler. The local profile is a repository containing personal information entities such as sex, religion, diet, etc. Entities have a schema that defines their nature. They are composed of: a name, a question, and possible answer. For example, the "sex" entity contains a question: "What is your gender?", and a possible answer of either "male" or "female". Entities are defined on the main server and downloaded by the client when needed (see Figure 5).

The client is responsible to query the consumer in order to populate the local profile repository. Whenever a mobile advertising-agent requires an entity that is not present in the local repository, the client will download the entity definition from the central server, after which the client will attempt to optimize the information retrieval. The more helpful the information is in computing affinity coefficients, the more urgent it is to acquire it. The agents' uncertainty values enable the client to extrapolate with a relatively high probability the most discriminating yet missing entity's answer. An unavailable entity's answer that is required by multiple mobile advertising-agents with high uncertainty values has higher discrimination power. The higher the discrimination power of an unanswered entity is, the more urgent is the retrieval of its answer.

Missing Information Handler. As previously mentioned, the matching estimator code handles missing information by increasing the uncertainty value. The uncertainty value returned by match estimator depends on the importance of the missing information needed to compute affinity coefficient. Missing information may have higher or lower impact in computing the match between the user and the advertisement depending on the importance of the information and how this information is used by the match estimator code. Thus, only the match estimator knows precisely how big the impact of missing information is, and only match estimator code can compute the uncertainty value. Match estimators of all non-expired agents with uncertainty values higher than 0 are periodically recomputed by the client. Every time the uncertainty value and affinity coefficient of an advertising-agent is recomputed, it gets a new opportunity to be selected and its advertisement has a chance to be displayed.

3.3 Mobile Advertising-Agents' Server

The server side is very light and does not require much computation since most of the logic is sent to the client and executed remotely. However it is the server's duty to verify the authenticity of the agents that will be delivered to the clients. To achieve this, advertisers have to cryptographically sign the agents. The server then verifies this signature. If verified, it adds its own signature and is ready to deliver the agent. Furthermore the server is responsible for limiting the client's workload. The server should limit the number of agents sent to the client, check for estimators codes that are too big to be executed, and make sure that a reasonable amount of information is stored on the client. We will further discuss this point in the discussion section.

In our implementation, the server is designed as a state-of-the-art RESTful web service provider. Fetching newly available mobile advertising-agent is a simple procedure. After client provides the identity (ID) of the last downloaded agent, the web service will assemble a list of non-expired agents with an ID higher than the recently downloaded one. For performance reason, we stream a list of agents instead of allowing one-to-one agent recuperation.

4 Implementation and Evaluation

The framework presented in this paper was implemented and tested on a large-scale field experiment. Based on our framework, we implemented a grocery discounts-discovery application for iPhone and made it available for free installation in the App Store of our country. The aims were to show the feasibility to execute targeting algorithms on modern smartphones like the iPhone, and to show the effectiveness of our algorithm in comparison with a random advertising approach.

To assemble the local information repository, we designed an ad hoc user interface, which is a full screen dialog window that asks profiling questions such as gender and lifestyle diet (see Figure 4 for a query example). Users have the

Fig. 4. The grocery discounts-discovery application. The main screen shows current discounts (left). Users can enter personal data, and are guaranteed that it will never be leaked from the phone (middle). Upon entering personal data, the user receives targeted discounts (right), and has the option to rate the discount in a five star scale.

possibility to either select the correct answer and save it, or skip the question. When a question is skipped three times, it is never asked again. A maximum of two profiling questions were asked per session[2].

To increase the usefulness of the application, we implemented a filtering and sorting function. The filtering functions allow users to filter discounts based on the grocery store, i.e., only the discounts in a specific grocery store are shown. The sorting function allows users to sort the discounts based on the ratings given by the users themselves (see Figure 4). The ratings are given in a one to five star scale.

To capture user interaction with the application, we implemented a logging system that locally saved important events such as application started, discount offer deleted, etc. The logging system also saved the events' timestamp for later transfer to our server.

Users were tracked using the iPhone unique identifier (UDID); a 40 bytes token that uniquely identifies each device. When users launched the application, the logs collected in the previous session together with the UDID were sent to the server for analysis. Note that the logging system was added for the field experiment purpose only. As it is not part of the conceptual framework it doesn't compromise user privacy there. We only logged the total number of events, i.e.,

[2] A session starts from when the user launches the application until he/she closes it.

the number of advertisements displayed, the number of advertisements deleted, the number of questions asked and skipped, etc.

4.1 Preliminary Conceptual Test

We ran a two-week pilot test with 10 participants, recruited among the undergraduate students of ETH Zürich. The pilot test was designed to check our concept and validate the functions of our prototype. In all, 509 mobile agents were transmitted, 275 discounts information were displayed, 45 discounts information were deleted, and 156 out of the 281 discounts information that were rated by the consumers had 3 stars or higher on a scale of 5. The 10 participants were surveyed after completing the test phase and all affirmed that none of the presented discounts information was off-target (e.g. a vegetarian did not receive a discount for meat). The participants also gave some comments on the usability of the application: 'it is useful to keep track of ads", 'it's easy to find out where the best discounts are, so you can choose at home where to shop', 'best offer in a glance', 'since the interface was nicely done it was never a problem for me to answer the profiling questions'.

4.2 Field Experiment

After the successful pilot test, we ran a field experiment. In collaboration with a local firm, we distributed real grocery discounts information through our application. Each week we distributed an average of 150 new discounts information of the three largest local grocery stores. We ran the experiment for 3 months with 903 participants. To evaluate our targeting mechanism, we incorporated a control group in the field experiment. The control group was equipped with a version of the application that randomly selected the discounts information without any targeting feature. We randomly assigned the consumers to the test or control group. It is important to note here that once a consumer was assigned to a particular group, he/she could not change it. The advertisement selection algorithm and iPhone identifier were stored in a DB. Even if they uninstalled and reinstalled the application, they would still get the same advertisement selection algorithm. 52% of the users (470) were assigned to the test group (with targeting algorithm) and 48% (433) to the control group (with the random selection algorithm).

Performance Analysis. The first analysis focuses on the performance of the proposed framework to demonstrate its robustness and feasibility of a concrete implementation. Our measurement units are sessions. We consider each session as the time from when the application is launched until it is closed. Sessions were retrieved from 470 users in the test group, i.e., those running the targeting algorithm. We selected 725 sessions for the analysis based on the consideration that at least one agent was downloaded, and the evaluation and selection phases were successfully completed. This is to make sure that we excluded those who closed the app immediately after launching it. We also conducted an anomaly

filtering, i.e., all sessions having unusual time duration for agents' downloads due to unexpected networks, hardware, or software behavior were excluded. The descriptive statistics of the 725 sessions are shown in the table below.

Table 2. Descriptive statistics of the sessions. User hardware: iPhone 3G/3GS.

	Min.	**Max.**	**Mean**	**Std. Dev.**
# of Agents Downloaded	1.00	50.00	40.13	15.85
Avg. Download Time (in sec.)	0.00	37.00	5.24	5.17
# of Carried Ads Displayed	1.00	71.00	11.03	6.96
Processing Time (in sec.)	0.00	48.00	3.03	8.55

The average number of the carried advertisements displayed per session is 11.03 and the average time to process the downloaded agent is 3.03 seconds. Hence approximately 0.27 seconds are needed to execute the match estimator code of each agent and decide if the agent's carried advertisement should be displayed to the consumer. The average time to download an agent is 0.13 seconds. By analyzing the collected log entries, we estimate that the total time to process an agent (download + code interpretation + selection) is around 0.4 seconds.

Usage Analysis. Both test and control groups used their respective application mostly during the weekdays. As shown in Figure 5, there is no significant difference in the time-of-the-day when the test and control groups launched their respective application. However when we compared the number of times the test group launched our targeted advertising application (c.f. the control group) with an independent sample t-test analysis, we found a significant difference in the launching frequency of the test and control groups (sig. 0.006). The test group who had the targeted advertising application launched the application 18.2% more as compared to the control group. On average the application was launched 11.7 times by the test group and 9.9 times by the control group.

Besides examining the application launch, we also examined four common events in the test and control groups, i.e., open the details of discount information (eventID 7), rate the discount information (eventID 10), and sort the discount information based on the ratings (eventID 200) or retailers (eventID 201). We found that besides clicking on the advertisements to see more details, users in both groups extensively used the app feature that allows them to sort the advertisements based on the retailers (see Figure 5). Most probably this is because they are loyal to a specific retailer, and were using this feature when they were inside the shop. We also noted that application launches (EventID 1) reach peeks at 9h, 12h, 17h and 21h, which indicates that users prefer to browse for offers during breaks and in their free time, and most preferably in the evening.

The effectiveness of our targeting algorithm is proportional to the amount of user information in the local repository. The more often the application was used, the more information was collected in the local repository, and thus the better

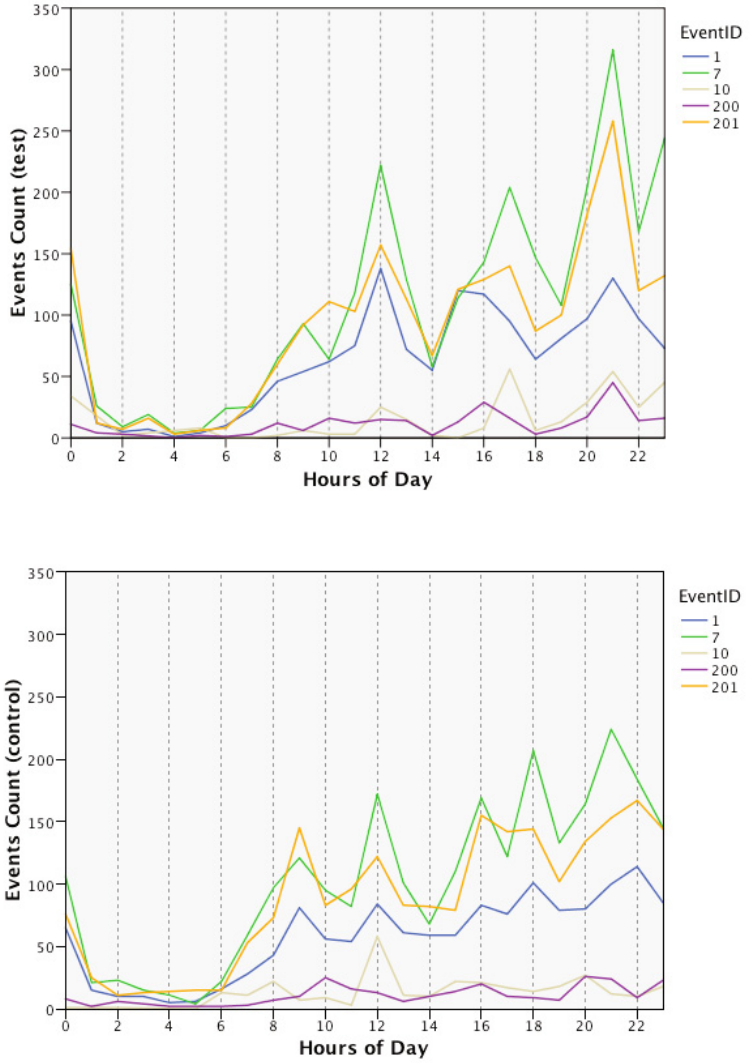


Fig. 5. Hourly user activity of the test (above) and control group (below). Events IDs: 1='app launched', 7='read discount', 10='rate discount', 200='sort discounts by rating', 201='sort discounts by retailer'.

the performance of our targeting algorithm should be. To gauge satisfaction with the displayed advertisements, we compared the average ratings given to the discounts information in the test and control groups. As shown in Figure 6, the test group had greater satisfaction with the displayed advertisements as they gave significantly higher ratings to the advertisements. The longer they used the targeted application, the higher the ratings they gave to the displayed advertisements. This confirms the effectiveness of our targeting algorithm.

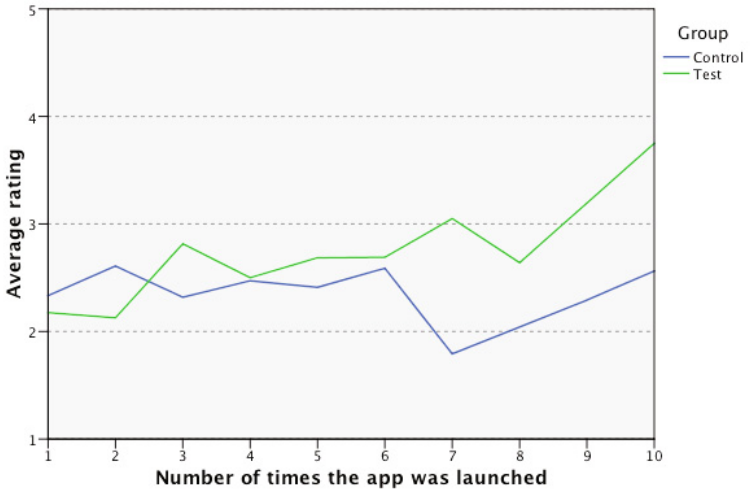


Fig. 6. Average ratings of the displayed advertisements, depending on the number app usages, for test and control group

5 Privacy and Threats

Having claimed that our targeting framework is privacy-preserving, in this section we argument on threats on user privacy, and how they are mitigated.

In our setting we define privacy the following way: no attacker should be able to derive any user profile data. Conceptually, our framework fulfills this property, because targeting is done on the user device itself rather than on a untrusted remote server, and that no user profile data is transmitted to any second party.

To break the privacy property, the goal of an attacker would be the following two things. 1): accessing user profile data on the user device and transmit it to the attacker or 2): deriving user profile data implicitly from user behavior.

Let's first consider the first point: One possibility to access the user data would be to install a malicious program on the user device that reads out the desired values and that transmits them to the attacker's server. Here we rely on the smartphone OS security policy (in our case iOS) and our configuration that disables read access from other apps to our app's data. Apple further tightly reviews any app before it can be published to the app store, which further diminishes this threat. Another potential threat comes from the JavaScript targeting codes that run on the user device and that have access to profile data. Here we rely on the security of the sandboxed Webkit instance that runs the JavaScript. If configured accordingly, any data transmission in the sandboxed environment to the external is blocked. Therefore, even if malicious JavaScripts were able to reach the client they would be unable to communicate any stolen information. In practice the sandboxed environment may experience vulnerabilities that allow to circumvent the security policies. Browser security and especially sandboxing

is however widely researched and implemented nowadays (e.g., [20]), so that we decide to rely on the Webkit implementation in our case.

The second point is mitigated because in our discount discovery application, no click data is transmitted to any server. In contrast to the presented case study, marketers however often require the advertising publishers to provide statistical feedbacks on the advertising campaign (E.g. impressions, clicks). In case of the presence of a statistical feedback loop, the customer privacy could theoretically be compromised. In theory a malicious server could estimate the customer profile by analyzing the client feedbacks. To mitigate probing attacks, security mechanisms for anonymizing requests such as anonymous ping, delay tolerant networks [12] and onion routing [10] can be combined to the presented framework. Passive eavesdropping attacks would so be circumvented. Hence successful attacks would have to actively alter the advert server's normal behavior. For example, if the server only sends the advertisement agents to one mobile client, he could track his clicks even if the clicks are anonymized.

Summarized, our implementation fulfills the proposed privacy policy. In a future implementation that enables to track clicks, the mechanisms described in this section should be implemented.

6 Discussion and Further Work

While offering numerous benefits, our proposed framework may encounter resource management problems when the match estimator codes are too complex or the repository is too big such that they may overload the mobile advertising-agents' client. In our field experiment, we overcame this potential issue by limiting the number of agents sent per session (max. 50 agents per session), limiting the code size of the agents (max. 1KB per agent), and meticulously defining which information should be collected and stored in the consumers' repository. While they may not be ideal measures, these three measures could prevent the resource management problem from happening.

A second limitation of our framework is that it may not be able to address special cases such as wrong match estimator codes and codes that never terminate (infinite loops). By now our assumption is that the server delivers agents with valid code. Nevertheless, our field experiment showed that the proposed framework has well-performed and such special cases could be prevented. In our field experiment, all agents' match estimator codes were computationally generated by assembling multiple previously verified pieces of codes.

The above limitations present opportunities for future research. Future research can also benefit from this study as it offers detailed guidelines on how to implement and field test a proposed framework. Such field test is currently lacking in most of the previous studies proposing targeted advertising methods.

In the field experiment we assessed the effectiveness of the targeting algorithm by using the framework in a grocery discounts-discovery context, we examined the application performance and usage, and compared consumer satisfaction with the targeted advertisements vis-à-vis randomly displayed advertisements.

Future research could further compare our mobile-advertising agent framework with other existing frameworks for targeted advertising. An indication for improved user-satisfaction with privacy-preserving targeting vis-à-vis non-privacy-preserving targeting is given in [23].

By addressing the security of the proposed framework two important aspect need to be mentioned: First, it is crucial that the targeting code runs in an environment that prevents any Internet connection and any profile modifications. We here rely on a sandboxed environment (i.e. Apple WebKit) that fulfills these requirements. So even malicious advertising agents are unable to transmit or alter any private information. Second, advertisers generally like to track the clicks of their advertising campaigns. In our current implementation this is not possible as this would leak information on who clicked on the ads. A way to overcome this limitation is to anonymize the click information. This can be achieved with onion routing [10], anonymous pings or delay tolerant networks [12]. Given that many threats on privacy arise because advertisers wish to track user clicks for payment reasons (pay-per-click), we finally highlight the need for further research on privacy-preserving ad-payment schemes.

7 Conclusion

In this paper, we presented a novel framework of agent-based targeted advertisement that allows for advertisement targeting while preserving users' privacy. We presented its concrete implementation with a grocery discounts-discovery application for iPhone, and evaluated it in a large-scale field experiment with 903 real users. Based on the field experiment, we analyzed among other things the application performance and user satisfaction. We showed that the total time needed to fetch and display an agent-based advertisement in our testing environment (iPhone 3G/3GS) is a mere 0.4 seconds. To assess the effectiveness of our targeting algorithm, we compared the average ratings given to the targeted advertisements vis-à-vis randomly displayed advertisements. We discovered that users were generally more satisfied with the targeted advertisements as reflected in the significantly higher ratings given to the targeted advertisements. Moreover, we discovered that the longer they used the targeted application, the higher the ratings they gave to the displayed advertisements. This paper demonstrates that mobile agents can be effectively used to achieve targeted advertisement. The computational overhead imposed by the mobile agents is minimal and therefore tolerated by end users. Furthermore, we demonstrated how mobile agents can be used to preserve user privacy. We presented a validated framework that can be used as base for future development of privacy-safe and context-aware mobile target content delivery.

References

1. Aaker, J., Brumbaugh, A., Grier, S.: Nontarget markets and viewer distinctiveness: The impact of target marketing on advertising attitudes. Journal of Consumer Psychology 9(3), 127–140 (2000)

2. Aalto, L., Göthlin, N., Korhonen, J., Ojala, T.: Bluetooth and wap push based location-aware mobile advertising system. In: 2nd International Conference on Mobile Systems, Applications and Services, pp. 49–58 (2004)
3. Aggarwal, P., Krishnaswamy, D., Daley, R.S., Lundqvist, P.: User profile generation architecture for mibile content - message targeting. U.S. Patent Application No.: 0319329A1 (20090319329) (December 2009)
4. Alt, F., Balz, M., Kristes, S., Shirazi, A., Mennenöh, J., Schmidt, A., Schröder, H., Goedicke, M.: Adaptive user profiles in pervasive advertising environments. In: Ambient Intelligence, pp. 276–286 (2009)
5. Altmeyer, C.: Smartphones, social networks to boost mobile advertising (June 2009), http://www.reuters.com/article/idUSTRE55S2FY20090629
6. Barnes, J., Distler, P., McMullen, M., Sharma, S.: Architecture for mobile advertising with location. US Patent Application No.:0227467A1 (March 2007)
7. Bulander, R., Decker, M., Schiefer, G., Kolmel, B.: Comparison of different approaches for mobile advertising. In: 2nd IEEE International Workshop on Mobile Commerce and Services, pp. 174–182 (July 2005)
8. Farmer, W., Guttman, J., Swarup, V.: Security for mobile agents: Issues and requirements. In: 19th National Information Systems Security Conference, vol. 2, pp. 591–597 (1996)
9. Gao, J.Z., Ji, A.: Smartmobile-ad: An intelligent mobile advertising system. In: 3rd International Conference on Grid and Pervasive Computing Workshops, pp. 164–171 (2008)
10. Goldschlag, D., Reed, M., Syverson, P.: Onion routing. Communications of the ACM 42(2), 39–41 (1999)
11. Guha, S., Reznichenko, A., Tang, K., Haddadi, H., Francis, P.: Serving ads from localhost for performance, privacy, and profit. In: 8th Workshop on Hot Topics in Networks (2009)
12. Haddadi, H., Hui, P., Brown, I.: Mobiad: private and scalable mobile advertising. In: 5th ACM International Workshop on Mobility in the Evolving Internet Architecture, pp. 33–38 (2010)
13. Langheinrich, M.: Personal privacy in ubiquitous computing. Diss., ETH Zürich, Nr. 16100 (2005)
14. Mahmoud, Q., Al-Masri, E., Wang, Z.: Design and implementation of a smart system for personalization and accurate selection of mobile services. Requirements Engineering 12(4), 221–230 (2007)
15. Masinter, L.: The "data" url scheme. IETF (August 1998), http://tools.ietf.org/html/rfc2397
16. Narayanaswami, C., Coffman, D., Lee, M., Moon, Y., Han, J., Jang, H., McFaddin, S., Paik, Y., Kim, J., Lee, J., et al.: Pervasive symbiotic advertising. In: 9th Workshop on Mobile Computing Systems and Applications, pp. 25–26 (February 2008)
17. Perdikeas, M., Chatzipapadopoulos, F., Venieris, I., Marino, G.: Mobile agent standards and available platforms. Computer Networks 31(19), 1999–2016 (1999)
18. Pham, V., Karmouch, A.: Mobile software agents: an overview. IEEE Communications Magazine 36(7), 26–37 (1998)
19. Ranganathan, A., Campbell, R.: Advertising in a pervasive computing environment. In: 2nd International Workshop on Mobile Commerce, pp. 10–14 (2002)
20. Reis, C., Barth, A., Pizano, C.: Browser security: lessons from google chrome. Communications of the ACM 52(8), 45–49 (2009)
21. Shannon, R., Stabeler, M., Quigley, A., Nixon, P.: Profiling and targeting opportunities in pervasive advertising. In: 1st Workshop on Pervasive Advertising at Pervasive (2009)

22. Specification, E.: Standard ecma-262. ECMA Standardizing Information and Communication Systems (1999)
23. Sutanto, J., Palme, E., Tan, C.H., Phang, C.W.: Addressing the personalization-privacy paradox: An empirical assessment from a field experiment on smartphone users (unpublished working paper, 2012)
24. Toubiana, V., Narayanan, A., Boneh, D., Nissenbaum, H., Barocas, S.: Adnostic: Privacy preserving targeted advertising. In: 17th Annual Network & Distributed System Security Symposium (2010)
25. Yang, W.S., Cheng, H.C., Dia, J.B.: A location-aware recommender system for mobile shopping environments. Expert Systems with Applications 34(1), 437–445 (2008)

Towards Timely and Efficient Semantic Reasoning for the Networked Society

Bo Xing[1], Johan Hjelm[2], Takeshi Matsumura[2], Shingo Murakami[2], Toshikane Oda[2], and Andrew Ton[1]

[1] Ericsson Research, USA, 200 Holger Way, San Jose, CA 95134, USA
[2] Ericsson Research, Japan, Momento Shiodome 2-3-17, Higashi-Shinbashi, Minato-ku, Tokyo 105-0021, Japan

Abstract. This paper presents our work in progress on enabling computerized reasoning capability in machine-to-machine communication scenarios for the Networked Society (or Internet of Things). Such reasoning capability is about drawing high-level conclusions on the situation in real time based on raw data streams generated by various sources. There are challenges posed by the dynamic and heterogenous availability of raw data coming from different sources, as well as the stringent time constraints for conclusions to be made. Our goal hence is to make machine-based reasoning processes time-efficient, resource-efficient, and scalable. We present an approach that addresses the challenges by decomposing a reasoning process into two stages: "shallow reasoning" and "deep reasoning". The former deals with the dynamic and heterogenous availability of raw data from different sources, while the latter executes semantic reasoning with a lightweight workload that has been reduced by the former. We present our prototype implementation of a reasoning system that adopts the proposed approach in a proactive healthcare use case. Performance evaluation is currently ongoing to verify the effectiveness of our approach.

Keywords: Reasoning, Semantic, Ontology, Timeliness, Efficiency.

1 Introduction

At Ericsson, we envision that, by the year of 2020, there will be 50 billion connected devices on this planet. By then, any device that would benefit from being connected will be connected (through any means). It is Ericsson's strong belief that when one person connects, their life changes; with everything connected, our world changes. This vision of "Networked Society" [1] (or "Internet of Things") is not about simply providing bit-pipes for people and their devices. It is rather to inter-connect the device network, the people network and the information network, and to have them autonomously and intelligently work for the good, facilitating us to collaborate, innovate, sustain, learn, care and participate.

To enable the needed autonomy and intelligence in the Networked Society, machine-based reasoning capability is an indispensable building piece. It is the

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 264–273, 2013.

capability of machines deriving high-level conclusions from low-level raw data sets. The conclusions facilitate decisions to be made for further actions to be taken without human intervention. In many cases, input data constantly comes from various data sources (e.g., machines or sensors) as streams and is brought together for machines to make sense of it, so that appropriate actions can be promptly and automatically taken by the same or other machines. For example, a public safety surveillance system comprising sensors and actuators reasons over contextual readings collected from the sensors to infer the situation, and based on that triggers the right actuators to respond to the situation accordingly. In another example, a health monitoring application detects anomalies in a person's health condition based on biometric measurements from personal health devices plus other contextual info, and reports it to the person's primary doctor.

Such Machine-to-Machine (M2M) communication, processing and actuation scenarios pose a set of challenges on a reasoning system. First, the reasoning mechanisms available today (such as fuzzy logic, Bayesian logic, case-based reasoning, and semantic reasoning) were designed for different purposes and hence come with varying resource consumption profiles (in terms of processing power usage, processing time and memory footprint). How to combine and make the best use of these techniques while optimizing overall resource consumption needs to be explored. Second, on the input end of a reasoning system, availability of data at various sources is not always deterministic. Inconsistent update frequency at data sources, broken connectivity to data sources and others can all cause unavailability, intermittent availability or delayed availability. Third, on the output end, the receivers of the conclusions from the reasoning system oftentimes have stringent requirements on timeliness - they issue real-time queries/requests and need to get back the answer before a deadline. Hence, they typically are okay with a coarse-grained answer as long as it meets the deadline. In case of less stringent time constraints, they certainly would like the answer to be as fine-grained as possible. Fourth, with millions and billions connected things in the "Networked Society", a reasoning system will need to deal with a large number of reasoning tasks concurrently. Hence, making it scalable is of great importance. Last but not the least, the reasoning system desirably comes with semantic support for increased interoperability, meaning that the conclusions it draws are made available in machine-understandable forms, enabling more sophisticated events to be auto-triggered and the corresponding actions to be auto-taken.

In this paper, we develop a reasoning framework that aims to address to the maximum extent the aforementioned challenges, namely resource efficiency, data availability, timeliness, scalability and semantic support. In brief, our approach is to constitute the core of a reasoning system with "*Shallow Reasoners*" and "*Deep Reasoners*". Deep reasoners employ semantic web technologies to enable interoperability and semantic support, and hence take the relatively heavy-weight reasoning workload. On the other hand, shallow reasoners are intended to offload part of the reasoning work (as a first attempt to enhance resource efficiency) as well as to deal with data availability at various sources. Thus, a reasoning process becomes a pipelined 2-stage procedure. In the first stage - shallow reasoning,

shallow reasoners each handling data from a particular source pre-processes the data and generates intermediate results that will facilitate semantic reasoning. In the second stage - deep reasoning, deep reasoners takes over from shallow reasoners the intermediate results as they become available, and incrementally performs ontology-based reasoning over them; they make available just-in-time conclusions with increasing granularity. Deep reasoners are optimized to minimize resource consumption and concurrently accommodate as many reasoning tasks as possible. As a proof of concept, we have implemented a prototype reasoning system that is targeted for a specific use case scenario - proactive healthcare.

The remainder of the paper is organized as follows. We first review related prior work in Section 2. We then present our approach to reasoning and the rationale behind it in Section 3. Following that, Section 4 describes our implementation of a prototype system that adopts the proposed approach to address a real-world use case. We finally conclude the paper in Section 5 by briefing our plan on performance evaluation.

2 Related Work

The vision that we are moving towards the Internet of Things era has been shared within the research and industrial communities [2] [3]. A large body of studies have been done and continue to be conducted on how to make the evolution happen [4] [5]. The first thing clearly is to make physical things connected, communicate and be recognizable, and a significant amount of efforts have made that possible today through various types of technologies [6] [7]. That however does not suffice to make a Networked Society, as intelligence needs to be brought to the connected, recognizable and communicating things for them to be able to work with people, for people and on behalf of people. A popular approach towards enabling this intelligence is to fuse semantic web technologies into connected sensors and objects, so that the information generated by any of them is understood in a common and consistent way [8] [9]. With that, sharable knowledge is created through the use of semantic reasoning techniques [10] [11].

3 Our Approach to Reasoning

One simple way of reasoning is to use pre-programmed reasoning rules, as is typically implemented in database applications. Such rules allow for deriving conclusions based on specific data - but only on those data. Including a new data type requires re-programming. In the "Networked Society" scenario, data sources are not static, and new data sources can be added and existing data sources removed in an ad-hoc manner. Considering that, a more favorable approach to reasoning is to have heterogeneous data structured in a machine-understandable way, which allows conclusions to be drawn automatically by a piece of software, often assisted by a set of generic rules. Thus, there are no longer limitations on which data sources can be used and which problem spaces can be addressed.

In light of the above, we choose to represent data using ontology. An ontology formally represents knowledge as a set of concepts within a domain, and the relations between these concepts [12]. By using ontology, the explicit specification of a problem domain can be shared, thus enabling interoperability between the reasoning system and other systems, services and applications. The use of ontology also makes it easy to introduce new problem spaces to the reasoning system (by simply importing ontologies that capture different domains). Moreover, with the expressiveness of ontology, supporting sophisticated reasoning, processing and querying with semantics is possible.

3.1 Overview

Ontology-based reasoning, however, is a non-trivial and costly job at run-time – it is highly time- and resource-consuming [13]. Even though the ontology used is kept as small as possible, long reasoning time and high resource consumption is still an unavoidable price to pay. To minimize the negative impact that ontology-based reasoning has on performance while still harnessing its power in sophisticated reasoning and semantic support, we need to (1) optimize the size (and possibly the shape) of the *base ontology* that describes the reasoning problem domain, (2) use ontology-based reasoning only for the part of the reasoning work that needs its power in semantic reasoning, and (3) decouple ontology reasoning from data sources, which may come and go dynamically and have availability constraints.

To that end, we approach reasoning with two key mechanisms. First, we define the base ontology to include only high-level concepts and to entail a conclusion tree. Without low-level concepts (such as measurements at sensory machines), the base ontology significantly shrinks in size. The conclusion tree contains possible conclusions at different levels of granularity, thus allowing for timely exposure of conclusions in an incremental manner. Second, we conduct reasoning as a two-step process – "shallow reasoning" and "deep reasoning". Shallow reasoning is the process of reasoning over raw data from data sources and deriving intermediate results that are representable in the base ontology. Deep reasoning is the process of reasoning over the base ontology containing the intermediate results and drawing conclusions. Working together, shallow reasoning offloads the part of work that is simple but deals with large amount of data (probably with availability constraints), while deep reasoning makes the most efficient use of the expensive but powerful semantic reasoning techniques.

3.2 Base Ontology – Minimizing Reasoning Complexity and Enabling Timely Availability of Conclusions

For minimizing its size, we define the base ontology such that it does not include low-level concepts such as measurements at sensory machines, but only high-level concepts like what a measurement indicates. To enable timely availability of conclusions, it conceptually entails a conclusion tree.

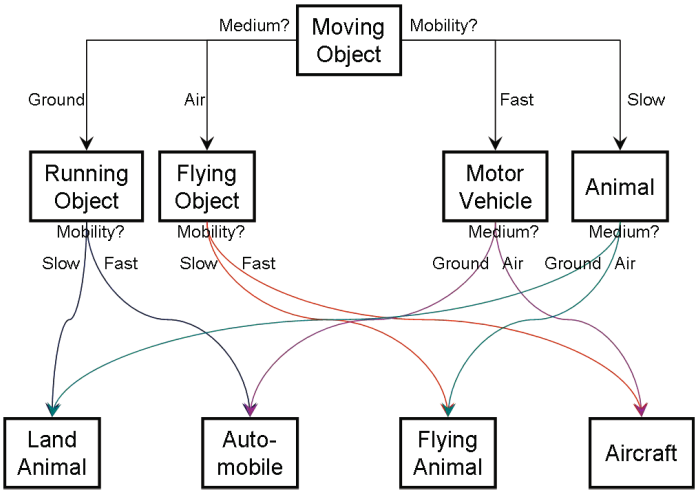


Fig. 1. An Illustrative Example of a Conclusion Tree

The nodes in a conclusion tree represent possible conclusions the reasoning system may arrive at. Fig. 1 illustrates an example conclusion tree in the case of a surveillance reasoning system, where the goal is to detect the type of a moving object based on sensors that identify the speed of the object and the medium it moves in. The root node of conclusion tree is the known concept that all possible conclusions belong to (e.g., an moving object in the case of a surveillance system). Every other node represents a conclusion that is refined over the conclusions represented by its parent nodes (note that here each node can have multiple parent nodes, and hence the conclusion tree is not a tree in the graph theory sense).

The refinement of a conclusion to reach a child node in the conclusion tree is enabled by the availability of additional data from data sources, and is formally stated as a rule (or Equivalent Class property) in the base ontology. In our example, when Medium data is available, a conclusion of Animal can be refined to become either Land Animal or Flying Animal. Further, the refinement can be accompanied by a confidence value or a formula to calculate it (e.g., defined as a SWRL (Semantic Web Rule Language) rule), indicating the accuracy or certainty of reaching the refined conclusion. The depth of a node in the tree implies the granularity of the corresponding conclusion - the deeper a node is at, the finer-grained the conclusion is; a leaf node represents a conclusion that has the finest granularity possible. In our example, the four conclusions at the bottom of the tree are the best conclusions we may end up drawing.

3.3 Deep Reasoning – Further Reducing Resource Consumption and Enhancing Scalability

In addition to exploiting shallow reasoning for lower resource consumption, optimizations are employed inside the deep reasoning process to further reduce the footprint. Although the conclusions from deep reasoning are persisted in an ontology database, the process of deep reasoning is executed in memory. This is in light of the poor performance of executing ontology reasoning over ontology databases.

The deep reasoning process manages each reasoning task as an ontology graph comprising instances of the concepts defined in the base ontology. Taking the surveillance system depicted in Fig. 1 as an example, a reasoning task handles detecting the type of one particular moving object. Hence, its ontology graph would contain instances of concepts including the mobility of the particular moving object and the medium it moves in, which are instantiated from intermediate results generated by shallow reasoners once they become available. Whenever such intermediate results are received from shallow reasoners, ontology-based reasoning is executed over ontology graphs to derive possible conclusions.

For an ontology graph, as soon as a conclusion that is not the root node of the conclusion tree is arrived at, the inferred ontology is persisted in the ontology database. Thus, the conclusion is immediately available for the receivers of outcomes from the reasoning system. For example, the conclusion that the moving object is a motor vehicle is good enough for some actions to be taken by the surveillance system; later when the conclusion is refined to be an aircraft, the surveillance system could finetune its reaction. Finally, once a conclusion that is the leaf node of the conclusion tree is arrived at, its ontology graph is removed after being persisted. The reasoning task finishes, as the best possible conclusion has been drawn.

In order to enhance scalability, each run of ontology-based reasoning works over a set of ontology graphs with an upper-bounded number of axioms (an axiom is a statement in an ontology). This is to account for the fact that the performance of ontology reasoning degrades exponentially with the increase in ontological axioms. When there are a large number of ontology graphs to be reasoned over, they are split up and assigned to different runs. Those runs can take place either sequentially or concurrently (if distributed to multiple deep reasoners physically residing on different machines).

4 Prototype Implementation

In this proof-of-concept prototype, we target a simple use case, where a user's situation (such as relaxing, dancing, drinking, mountain hiking, accident on freeway, etc.) is continuously derived in real time based on data collected from various sources. Such reasoning capability would facilitate recommendation, advertisement or event triggering services to become more personalized and context-aware. The data sources for the prototype include (but are not restricted to): (1) a fitness wrist band the user wears, (2) a smart phone the user carries, and

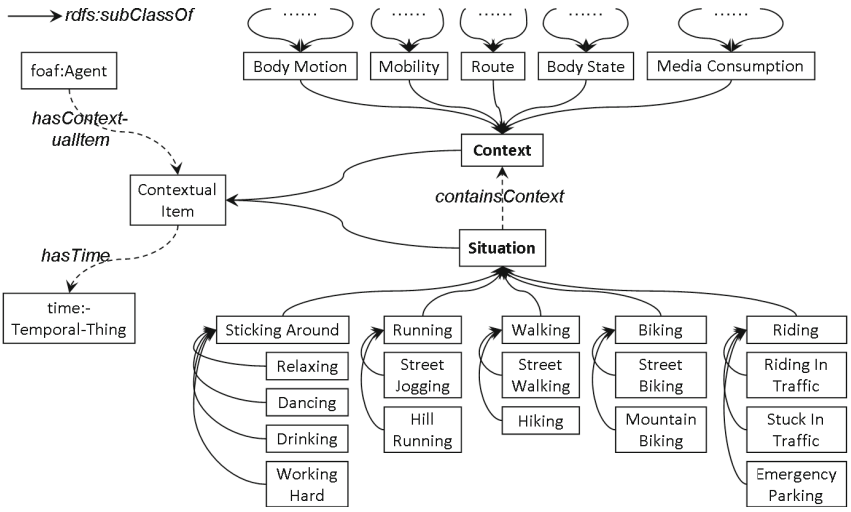


Fig. 2. Visualization of the Base Ontology

(3) an online music service the user subscribes to. Types of data available from these sources are (1) Galvanic Skin Response (GSR), (2) skin temperature, (3) wrist moving intensity, (4) GPS coordinates, and (5) music listened to.

4.1 Base Ontology

Fig. 2 is a visualization of the base ontology that defines our targeted problem. As we are concerned with deriving a user's situation during a particular period of time, the most important class in the ontology is the "Situation" class. That is the class which all conclusions to be drawn are instantiated as individuals as. Another important class is the "Context" class, which defines possible intermediate results that may come from Shallow Reasoners inferred from raw data. The direct subclasses of "Context" are the types of the intermediate results: "BodyMotion", "Mobility", etc. Further down this hierarchy are the possible values of each type of intermediate results, defined as their subclasses. For example, under "BodyMotion", there are "SignificantBodyMotion", "MediumBodyMotion", "InsignificantBodyMotion". Between "Situation" and "Context", there is an object property indicating that a "Situation" contains a "Context". This captures the fact that a situation comprises multiple contextual elements - they determine what situation it is. It is also the basis for ontology reasoning in our prototype. Under the "Situation" class, all possible conclusions are defined as the subclasses of "Situation" in a hierarchical way, forming a conclusion tree. Each subclass is defined with an equivalent class property, describing what "Contexts" the "Situation" should contain. For example, the "Relaxing" class is an equivalent class of "Situation" and "containsContext StaticMobility" and "containsContext InsignificantBodyMotion" and "containsContext CalmBodyState".

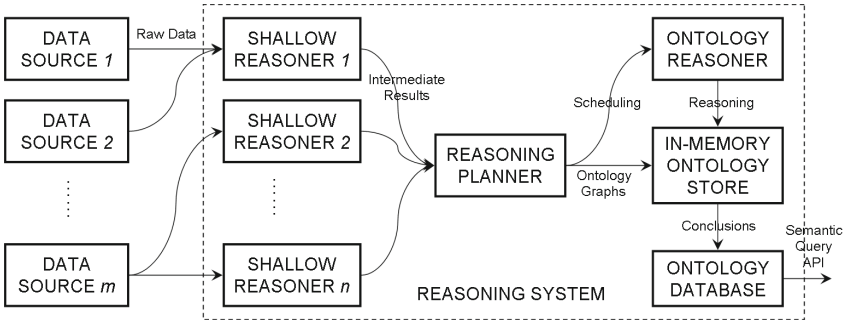


Fig. 3. Prototype Implementation Architecture

4.2 Architecture

The high-level architecture of the prototype is depicted in Fig. 3 (note that the components are logical, and physically can be distributed when needed). A set of Shallow Reasoners are connected to the corresponding data sources, preprocessing incoming raw data and deriving intermediate results. Each Shallow Reasoner connects to one or multiple data sources, and generates one or more intermediate results. They may employ various reasoning techniques such as fuzzy logic, Bayesian networks, etc. For our targeted use case in particular, we have a Body Movement reasoner, a Body State reasoner, a Mobility reasoner, and a Music Emotion reasoner. These Shallow Reasoners either use pre-programmed rules or exploit fuzzy logic reasoning through the use of the FuzzyDL library [14]. The output ends of the Shallow Reasoners are connected to the Reasoning Planner, which is responsible for scheduling and keeping track of reasoning tasks. The In-Memory Ontology Store is where the ontology graphs are dynamically loaded and ontology reasoning takes place; it is implemented using the Manchester OWL-API [15]. We choose to use Pellet [16] as the Ontology Reasoner, considering its incremental classification capability, compatibility with OWL-API, and support for SWRL rules. The Ontology Database is where the final conclusions (in the form of ontologies) are persisted and exposed to the outside world through semantic queries (e.g. SPARQL) or non-semantic queries (e.g., REST API parameters). We choose to use graph database Neo4j [17] as the base to develop our Ontology Database, and optimize it for potentially popular queries in our target scenarios.

Upon receiving an intermediate result from a Shallow Reasoner, the Reasoning Planner instantiates an individual of the "Context" class capturing it. If not done yet, it creates an ontology graph for the concerned situation by instantiating a "Situation" individual. The "Context" individual is filled in the ontology graph based on its relations with the "Situation" individual and other properties. The Reasoning Planner keeps track of the unique ID of each ontology graph , and associates a timer with it, controlling its lifetime inside the In-Memory Ontology Store. The Ontology Reasoner is scheduled to execute periodically when

there are ontology graphs in the In-Memory Ontology Store. When executing, it reasons over the base ontology plus all ontology graphs using incremental consistency checking and classification. It finds out the most specific subclass each "Situation" individual belongs to. If for a particular ontology graph, the "Situation" individual is derived to be also an individual of a subclass of "Situation", the inferred ontology graph is immediately persisted in the Ontology Database. Thus, the currently best available conclusion (the finest-grained type of situation) is made available for access right away. Meanwhile, if a finest-grained conclusion (a subclass of "Situation" that does not have any subclass) has been reached, the ontology graph is removed from the store (otherwise it will be removed when its predefined lifetime expires). As additional raw data comes from data sources over time, the conclusions exposed at the Ontology Database keep being updated with increasingly finer granularity.

5 Concluding Remarks

In this paper, we have presented our approach to timely and efficient reasoning for the Networked Society, where connected things intelligently work with, for and on behalf of people. We also presented our prototype implementation that proves the concepts in a specific use case setting. We are currently working on evaluating the performance of our approach by experimenting with the prototype system. The key performance metrics we look at are CPU usage, memory consumption and reasoning time. We design and run experiments to compare our approach against a mechanism that places all the workload on semantic reasoners, and to compare it against a system without using the conclusion tree method. Further, driven by the specific use case that we target, we vary the number of users to assess the scalability of the approach. We look forward to reporting the results of the evaluations in the near future.

References

1. Ericsson, Networked society, `http://www.ericsson.com/networkedsociety/`
2. Gershenfeld, N., Krikorian, R., Cohen, D.: The internet of things. Scientific American 291(4) (2004)
3. SAP, Internet of things: An integral part of the future internet (2011), `http://services.future-internet.eu/images/1/16/A4_Things_Haller.pdf`
4. C. Associati, The evolution of internet of things (2011)
5. Uckelmann, D., Harrisson, M., Michahelles, F.: An architectural approach towards the future internet of things. Architecting the Internet of Things (2011)
6. Welbourne, E., Battle, L., Cole, G., Gould, K., Rector, K., Raymer, S., Balazinska, M., Borriello, G.: Building the internet of things using rfid: The rfid ecosystem experience. IEEE Internet Computing 13(3) (2009)
7. Kortuem, G., Kawsar, F., Sundramoorthy, V., Fitton, D.: Smart objects as building blocks for the internet of things. IEEE Internet Computing 14(1) (2010)
8. Sheth, A., Henson, C., Sahoo, S.: Semantic sensor web. IEEE Internet Computing 12(4) (2008)

9. Pfisterer, D., Römer, K., Bimschas, D., Kleine, O., Mietz, R., Truong, C., Hasemann, H., Kröller, A., Pagel, M., Hauswirth, M., Karnstedt, M., Leggieri, M., Passant, A., Richardson, R.: Spitfire: toward a semantic web of things. IEEE Communications Magazine 49(11) (2011)
10. Wei, W., Barnaghi, P.: Semantic Annotation and Reasoning for Sensor Data. In: Barnaghi, P., Moessner, K., Presser, M., Meissner, S. (eds.) EuroSSC 2009. LNCS, vol. 5741, pp. 66–76. Springer, Heidelberg (2009)
11. Steller, L.A., Krishnaswamy, S., Gaber, M.M.: Enabling scalable semantic reasoning for mobile services. International Journal on Semantic Web and Information Systems 5(2) (2009)
12. Gruber, T.: A translation approach to portable ontology specifications. Knowledge Acquisition 5(2)
13. Reichle, R., Wagner, M., Khan, M.U., Geihs, K., Lorenzo, J., Valla, M., Fra, C., Paspallis, N., Papadopoulos, G.A.: A Comprehensive Context Modeling Framework for Pervasive Computing Systems. In: Meier, R., Terzis, S. (eds.) DAIS 2008. LNCS, vol. 5053, pp. 281–295. Springer, Heidelberg (2008)
14. Fuzzydl, `http://gaia.isti.cnr.it/straccia/software/fuzzyDL/fuzzyDL.html`
15. Owl-api, `http://owlapi.sourceforge.net/`
16. Pellet, `http://clarkparsia.com/pellet/`
17. Neo4j, `http://neo4j.org/`

A Reference Architecture for Mobile Code Offload in Hostile Environments

Soumya Simanta[1], Kiryong Ha[2], Grace Lewis[1],
Ed Morris[1], and Mahadev Satyanarayanan[2]

[1] Software Engineering Institute
Pittsburgh, PA USA
[2] School of Computer Science
Carnegie Mellon University
Pittsburgh, PA USA
{ssimanta,glewis,ejm}@sei.cmu.edu,
{krha,satya}@cs.cmu.edu

Abstract. Handheld mobile technology is reaching first responders and soldiers in the field to aid in various tasks such as speech and image recognition, natural language processing, decision making, and mission planning. However, these applications are computation-intensive and we must consider that 1) mobile devices offer less computational power than a conventional desktop or server computer, 2) computation-intensive tasks take a heavy toll on battery power, and 3) networks in hostile environments such as those experienced by first responders and soldiers in the field are often unreliable and bandwidth is limited and inconsistent. While there has been considerable research in code offload to the cloud to enhance computation and battery life, most of this work assumes reliable connectivity between the mobile device and the cloud—an invalid assumption in hostile environments. This paper presents a reference architecture for mobile devices that exploits cloudlets—VM-based code offload elements that are in single-hop proximity to the mobile devices that they serve. Two implementations of this reference architecture are presented, along with an analysis of architecture tradeoffs.

Keywords: reference architecture, mobile architecture, mobile systems, code offload, virtual machines, cloud computing.

1 Introduction

The number of apps and mobile devices created specifically for use in hostile environments, such as those experienced by first responders operating in crisis situations and military personnel, continues to grow [1][2][3]. First responders and soldiers can use handheld devices to help with tasks such as speech and image recognition, natural language processing, decision making and mission planning. However, several obstacles impede achieving the needed capabilities. First, mobile devices offer less computational power than conventional desktop or server computers, and are therefore not an ideal computation platform for tasks such as natural language

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 274–293, 2013.

processing and complex decision making. Second, computation-intensive tasks, such as image recognition or even global positioning systems (GPS), take a heavy toll on battery power. Third, networks in hostile environments are often unreliable and bandwidth is limited and inconsistent [4].

To overcome some of these challenges, cyber-foraging is employed to leverage external resources to augment the capabilities of resource-limited mobile devices [5][6][7][8][9][10][11][12][13]. However, many cyber-foraging strategies rely on the conventional internet or on environments that tightly couple handheld applications and servers on which code is offloaded (see Section 6). Cyber-foraging therefore addresses the challenges of conserving battery power and limited computing power, but does not address the challenges of unreliable networks and dynamic environments.

This report presents a strategy to overcome the challenges of cyber-foraging for mobile platforms in hostile environments by using *cloudlets* — discoverable, localized, stateless servers running one or more virtual machines (VMs) on which mobile devices can offload expensive computation. Cloudlets enhance processing capacity and conserve battery power, and simultaneously provide ease of deployment in the field.

2 Cloudlets as Intermediate Offload Elements

Code offload from mobile devices to cloud environments is the topic of much recent research [14][15][16][17][18][19][20]. This is also an approach used commercially by applications such as Siri for voice recognition [21]. However, an underlying assumption in these approaches is connectivity to the cloud, which is not always available or reliable in hostile environments.

A high-level architecture for code offload in hostile environments is proposed in [22] and presented in Figure 1. This architecture inserts an intermediate layer between the central core (i.e., enterprise cloud) and the mobile devices. At the heart of this architecture is a large centralized core that could be implemented as one of Amazon's data centers or a private enterprise cloud. At the edges of this architecture are offload elements for mobile devices. These elements, or cloudlets, are dispersed and located close to the mobile devices they serve [23]. This architecture decreases latency by using a single-hop network and potentially lowers battery consumption by using WiFi or short-range radio instead of broadband wireless which typically consumes more energy [24][25].

A key attribute of this architecture is that the offload elements are stateless. A mobile device does not need to communicate with the core during an offload operation; it only needs to communicate with its closest offload element. Communication between the offload element and the core is only needed during setup and provisioning. Once an offload element is provisioned, it can work in disconnected mode. Adding or replacing an offload element involves little setup or configuration effort.

One approach to offload is VM synthesis [23][26]. In this approach, an application overlay is offloaded from the mobile device to a cloudlet. An application overlay represents the difference between a base VM with only an operating system installed and the same VM with the application installed. In effect, the mobile device becomes the vector by which the needed application is deployed to the field. This approach

takes advantage of properties of VMs that reduce hardware dependencies—the same VMs can operate on several hardware platforms, and a single hardware platform can support multiple VMs—to provide flexibility in highly volatile environments where it is difficult to assure the right hardware or OS platform for cyber-foraging.

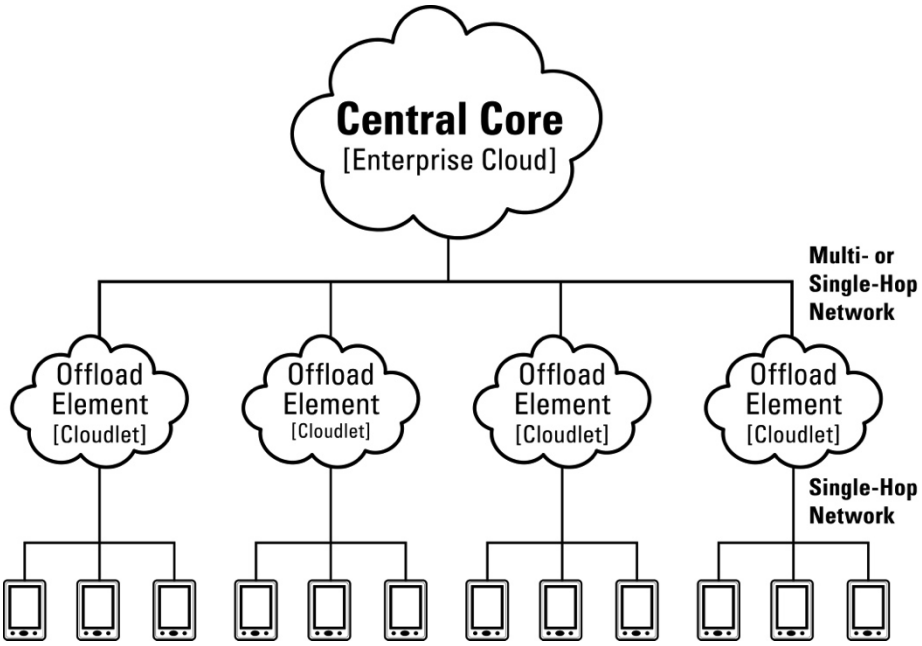


Fig. 1. Three-tier architecture for code offload

An application overlay is created once per application, as described in Figure 2. The Base VM is a VM disk image file that is obtained from the Central Core and saved to a cloudlet that runs a VM manager compatible with the Base VM. The VM manager starts the Base VM and the application is installed. After installation, the VM is shut down. A copy of the modified Base VM Disk Image is saved as the Complete VM Disk Image. The Application Overlay is calculated as the binary diff (VCDIFF rfc3284) between the Complete VM Disk Image and the Base VM Disk Image. The Base VM is then deployed to any platform that will serve as a cloudlet. The cloudlet may support multiple VMs, thereby reducing hardware dependencies on the offload element, and between the offload element and mobile device. A mobile device carrying application overlays will be able to execute these applications on any cloudlet that has the corresponding Base VM.

VM synthesis is particularly useful in hostile environments characterized by unreliable networks, loss of cyber foraging platforms, and a need for rapid deployment. For example, imagine a scenario where a first responder must execute a computation-intensive app configured to work with cloudlets. At runtime, the app discovers a

nearby cloudlet located in a rescue camp and offloads the computation-intensive portion of code to it. However, due to a natural disaster, loss of network connectivity, or exhaustion of energy sources on the cloudlet, the mobile app is disconnected from the cloudlet. The mobile app can locate a different cloudlet and have the app running in a short time, with no need for any configuration on the app or the cloudlet. This runtime flexibility enables the use of opportunistically available resources, replacement of lost cyber-foraging resources, and dynamic customization of newly-acquired cyber-foraging resources.

The following sections present the reference architecture for code offload in hostile environments and details of two prototype implementations.

3 Reference Architecture

Hostile environments are characterized by uncertainty in available resources such as computational capability and bandwidth. In addition, many applications that are useful in these environments are computation-intensive; these include face recognition, natural-language processing, route calculation, and text recognition, all of which require some form of input from sensors on the device. Soldiers and first responders executing missions are often away from their bases for many hours and cannot afford to carry many extra batteries. Therefore, a solution for code offload in hostile environments must consider 1) native apps that exploit device sensors, 2) code offload elements that can be quickly configured and deployed, and 3) battery consumption on the mobile device.

Figure 3 presents a reference architecture for mobile devices that exploit cloudlets for code offload. The major components of this architecture are the *Cloudlet Host* and the *Mobile Client.*

The *Cloudlet Host* is a physical server that hosts 1) a discovery service that broadcasts the cloudlet IP address and port to allow mobile devices to find it 2) the Base VM Image that is used for VM synthesis 3) a Cloudlet Server that handles code offload in the form of application overlays, performs VM synthesis and starts guest VM instances with the resulting VM images, and 4) a VM Manager that serves as a host for all guest VM instances that contain the computation-intensive server component of the corresponding mobile app.

The Mobile Client is a handheld or wearable device that hosts 1) the Cloudlet Client app that discovers cloudlets and uploads application overlays to the cloudlet and 2) a set of Cloudlet-Ready Apps that operate as clients of the server code running in the cloudlet. The Mobile Client stores an application overlay for each cloudlet-ready app that a user would conceivably want to execute and for which computation offloading is appropriate. Each application overlay is generated from the same Base VM Image that resides in the cloudlet. In an operational setting, these Base VM Images could be retrieved from the central core shown in Figure 1.

To validate the feasibility of the proposed reference architecture for hostile environments, we constructed a prototype as described in the next section.

4 Initial Prototype

The initial prototype is an implementation of a face recognition application in which the client is an Android app and a cloudlet-based server that contains computation-intensive code that performs face recognition. The client locates a cloudlet via a discovery protocol, sends the application overlay (Face Recognition Server code) to the cloudlet for VM synthesis and captures images and sends them to the Face Recognition Server on the cloudlet. This initial prototype was created based on the reference architecture presented in Figure 3. The prototype architecture is shown in Figure 4.

4.1 Base VM Image Creation

The Base VM Image for this prototype is a 3GB image with Microsoft XP plus the necessary system updates. To keep the image size small, "system restore" was turned off and unnecessary system components were removed using the DiskCleanup utility. Additional components were included in the Base VM Image to enable communication between the Guest VM and the *Cloudlet Server*. This additional complexity was one of the reasons for revising the initial prototype (see Section 5).

A major difference between the prototype and the cloudlet work presented in [23] is the type of client involved. Satyanaranyan's work uses a virtual network computing (VNC) client that acts as a remote desktop for the VM [27]. In our prototype, the Face Recognition Client is a rich client (native app) that interacts with a Face Recognition Server inside the VM. The Face Recognition Client requires the IP address and port of the Face Recognition server to establish a network connection. However, the Cloudlet Server does not directly know the IP address and port of the synthesized VM because the IP address is assigned by the Dynamic Host Configuration Protocol (DHCP) server executing in bridged mode and the VM host has no visibility into that assignment. The Cloudlet Client uses an HTTP request to start the cloudlet setup, and expects an HTTP response from the Cloudlet Server that contains the IP address and port of the Face Recognition Server.

To solve this problem, a CloudletStartup Windows service (implemented in Python) is included in the Guest VM that performs three functions:

1. starts the Face Recognition Server in a separate thread
2. reads the IP address and port from the Face Recognition Server configuration and communicates it to the Cloudlet Server in an HTTP POST
3. sends a periodic heartbeat to the Cloudlet Server in an HTTP POST

After completion of these steps, the VM was shut down and the resulting image file was saved as the Base VM Image.

4.2 Tools for Overlay Creation

The overlay for the face recognition application was created by following the steps presented in Figure 2. The VM Manager was started using the Base VM Image, the Face Recognition Server was installed, and the VM shut down. The specific tools used to "Calculate Diff between Complete and Base VM Image" are

- xdelta3: open-source binary diff tool that generates a file as the difference between the Base VM Disk Image and the Complete VM Disk Image [29]

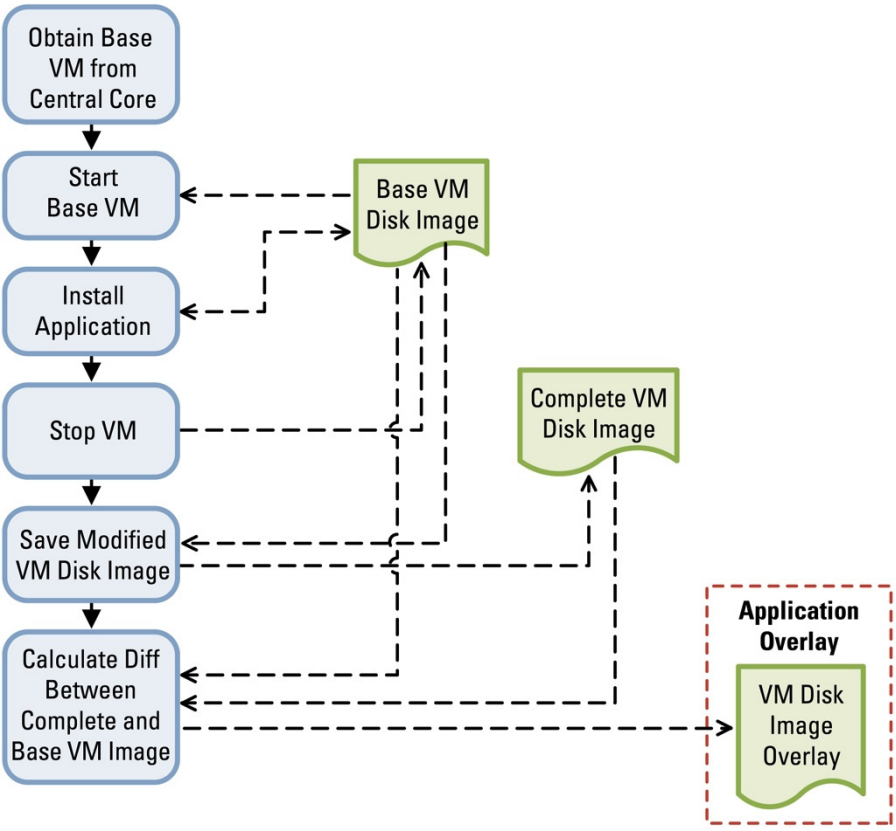


Fig. 2. Application overlay creation process

- lzma: set of public-domain libraries and tools that compress the diff file using the Lempel-Ziv-Markov chain data-compression algorithm [30]
- OpenSSL: open-source SSL (Secure Sockets Layer) implementation that encrypts the compressed file [31]

The resulting file is the application overlay. The Cloudlet Server uses these same tools in reverse order when it performs VM synthesis.

4.3 Cloudlet Host

The Cloudlet Host is an Ubuntu 10.10 Linux server that hosts the following sub-components.

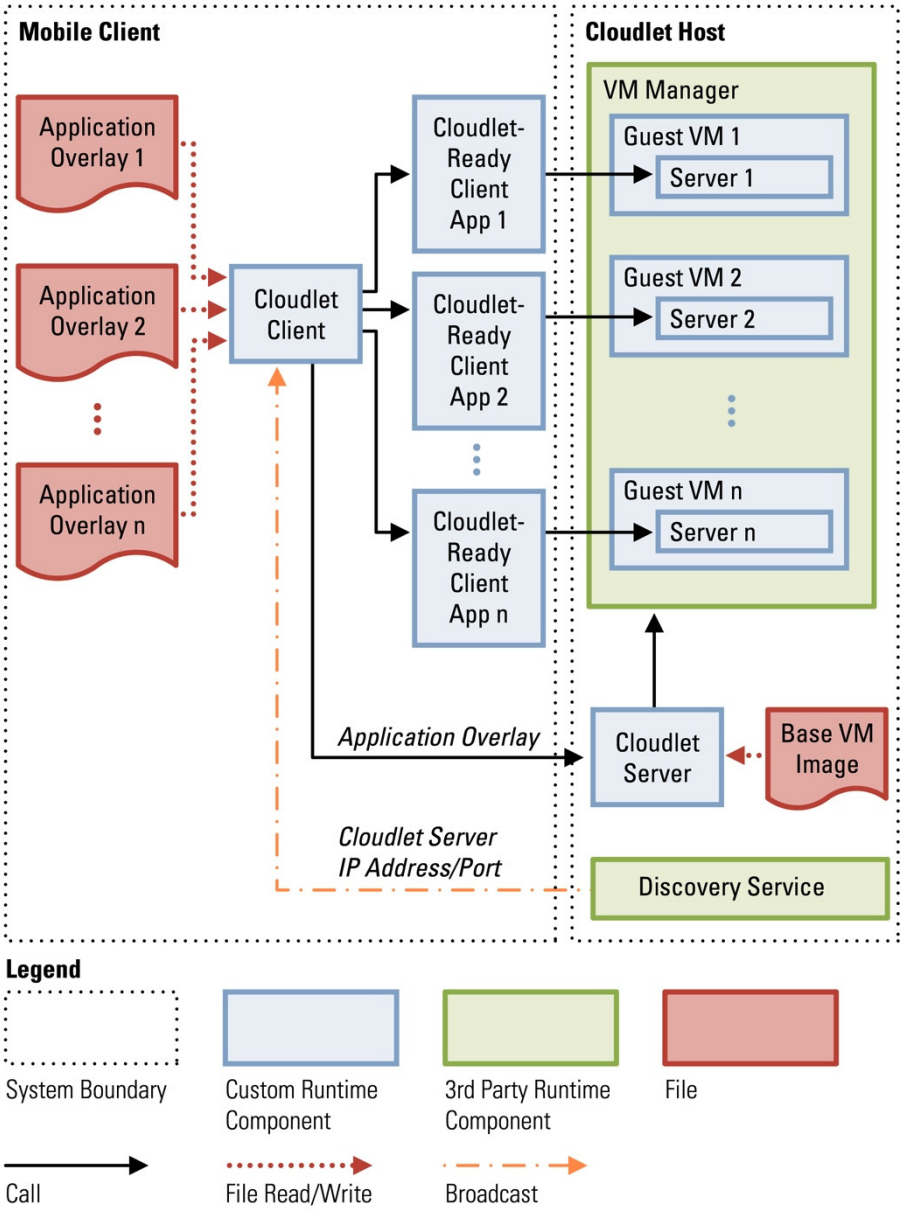


Fig. 3. Reference architecture for cloudlet-based code offload

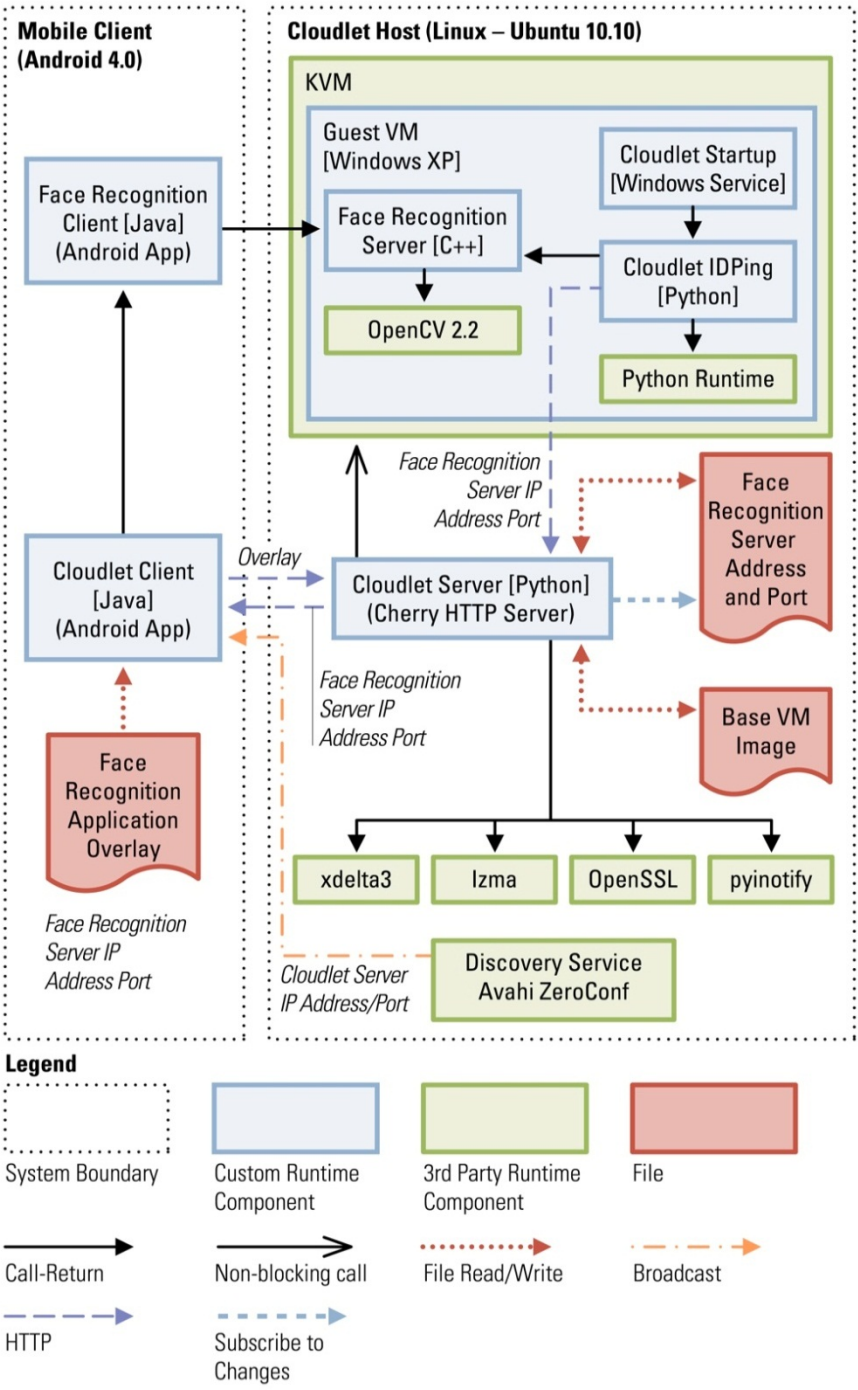


Fig. 4. Architecture for initial prototype

Kernel-Based Virtual Machine (KVM)

The virtualization infrastructure used in the prototype is KVM [32]. KVM is open-source and has good community support. KVM runs a Guest VM for each offloaded application. The Guest VM for the face recognition application is Windows XP, but KVM supports most popular operating systems [33].

Cloudlet Server

The core of the Cloudlet Server is an HTTP Server implemented using CherryPy, an extensible HTTP server [34]. The Cloudlet Client sends the encrypted and compressed overlay as an HTTP POST request. Upon receipt of the overlay, the overlay is decrypted and decompressed using the tools listed in Section B. VM synthesis is performed by using xdelta3 to apply the overlay to the Base VM Image and create the VM Image that contains the Face Recognition Server. The synthesized VM is started in bridged network mode, so that the Guest VM has a unique network-accessible IP address [35] and waits for notification that the Guest VM has successfully started. The Cloudlet Server uses pyinotify—a Python file system change monitoring utility—to subscribe to changes in the face recognition server IP address and port file [36]. On startup, the Guest VM executes the CloudletStartup Windows service (see Section A) to communicate the IP address and port back to the Cloudlet Server where another thread writes the IP and port information to a file. As soon as the file changes, the waiting thread in Step 3 receives notification. It reads the file and sends a response to the Cloudlet Client that contains the IP address and port on which the Face Recognition Server will be listening.

Discovery Service

The Discovery Service is based on the Avahi implementation of Zero Configuration Networking (ZeroConf) [37]. Zeroconf is a local network-discovery protocol for creating an IP network [38]. The Discovery Service broadcasts the Cloudlet Server IP address and port.

4.4 Cloudlet Client

The Cloudlet Client is an Android app that

1. Discovers cloudlets through information broadcast by the Discovery Service residing in the Cloudlet Host
2. Creates a HTTP connection to the Cloudlet Server for overlay transmission and uploads the overlay
3. Obtains the IP address and port that the Face Recognition Server is listening on from the Cloudlet server
4. Communicates the IP address and port of the Face Recognition Server to the Face Recognition Client app using a shared local file
5. Launches the Face Recognition Client.

4.5 Face Recognition Client

The Face Recognition Client is an Android app that executes in client/server mode with the Face Recognition Server running in the Guest VM. On launch, the Face Recognition Client reads the local file that contains the IP address and port of the Face Recognition Server and opens a TCP/IP connection to it. Images that the camera captures are sent to the Face Recognition Server.

4.6 Face Recognition Server

The Face Recognition Server is implemented in C++ using the OpenCV image recognition library that supports training or recognition modes [39]. When in recognition mode, it returns coordinates for the recognized faces plus a measure of confidence.

4.7 Prototype Evaluation

We evaluated the prototype using the previously described face recognition application and two additional computation-intensive applications that are representative of capabilities needed in hostile environments. Each application has an Android app (client) and a server corresponding to the computation-intensive offloaded code. The server-side applications are described below.

- OBJECT: Linux C++ application based on the CMU MOPED object recognition libraries [40]
- FACE: Windows XP C++ face recognition application described in Section 0
- SPEECH: Windows XP Java application based on the CMU Sphinx-4 speech recognition toolkit [41]

In addition, an overlay corresponding to a NULL application (VM simply started and stopped) serves as a baseline for the analysis of transmission overhead and battery consumption. Table 1 displays the application data.

Table 1. Application Data for Initial Prototype

Appl.	Platform	Lang.	Appl. Size (MB)	Base VM Disk Image (MB)	Compressed VM Disk Image Overlay (MB)
OBJECT	Linux	C++	27.50	3546	165.32
FACE	Windows XP	C++	17.65	3073	43.55
SPEECH	Linux	Java	51.04	3546	176.23
NULL	Linux	N/A	N/A	3546	0.12

We conducted all experiments using the configuration shown in Figure 5. We measured energy usage with a Monsoon Solutions Power Monitor and the corresponding Power Tool software [42]. To ensure good experimental control, interactive inputs were scripted.

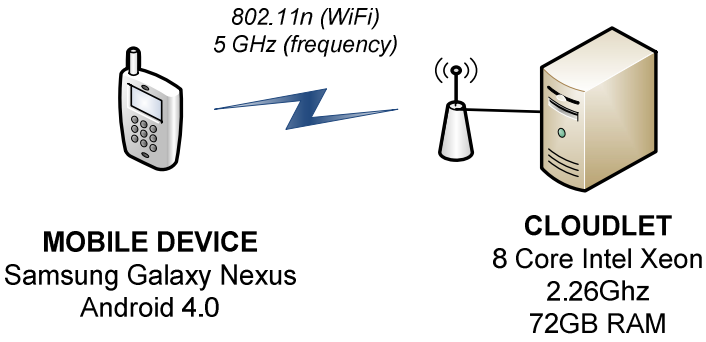


Fig. 5. Evaluation infrastructure setup

Average times for each step of the process and average energy consumption are shown in Figure 6. All times are measured from the client perspective and include HTTP Request/Response time.

The largest amount of time is consumed by the Upload Overlay operation and depends on overlay size (Figure 6). VM Synthesis and Start VM almost equally consume the second-largest time. VM Synthesis time is smaller for FACE because the sum of the base VM size and overlay size is smaller. FACE is also the outlier for Start VM because it is the only application that runs on Windows and therefore has a longer boot time. Energy consumption depends largely on overlay size. Average application ready time (time between upload overlay and Start VM) is between 101 and 166 seconds for the non-null applications. Given the dependence of these numbers on base VM image size and application overlay size, reducing the size of these files would reduce both application ready time and energy consumption. File size and implementation complexity are addressed in a revised prototype described in the next section.

5 Revised Prototype

In revising the initial prototype, our goals were first, to reduce application complexity and dependencies, and second, to decrease overall application ready time. The revised architecture shown in Figure 7 uses the same implementation of face recognition as for the initial prototype.

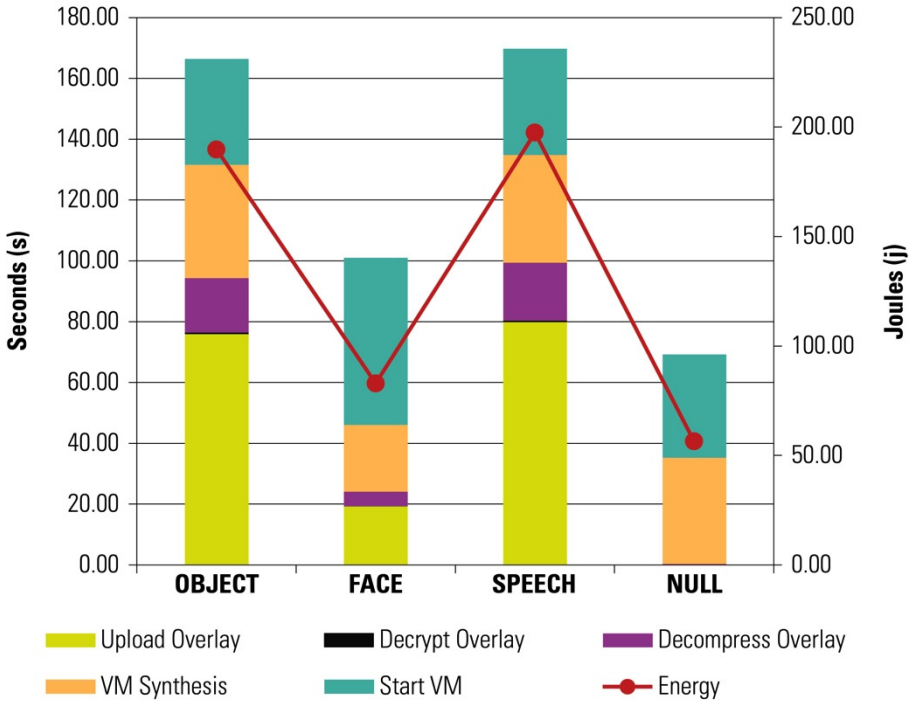


Fig. 6. Evaluation infrastructure setup

5.1 Main Changes

To address limitations of the initial prototype, we made the following changes.

Disk Image Format

The default format for KVM disk images is raw images. Storage for raw images is allocated during creation of the image. If a 3GB image is created, the VM manager writes data inside the image file that is fixed in size. This results in large pre-allocated images.

Another disk-image format that KVM supports is QEMU copy on write 2 (qcow2). The advantage of qcow2 is that storage allocation is delayed until actually needed. This means both size of the disk and overlay will be optimized. A disadvantage of qcow2 is the overhead caused by storage allocation at runtime. Reduction in the amount of time for overlay transmission made this tradeoff acceptable for the revised prototype.

Memory Snapshot Overlay Plus Disk Image Overlay

The initial prototype only transfers the disk image overlay. This means that the VM is always cold started and requires application-specific scripts to start the application and

send connection information back to the client, as explained in Section 4. In the revised prototype, a memory snapshot is also created. The three files (Base VM Disk Image, Base Memory Snapshot, and Base Disk Snapshot) now constitute the base VM image. When the VM is started, as shown in Figure 8, Base Memory Snapshot and Base Disk Snapshot are immediately applied. The application is installed, the VM is suspended, and a second set of snapshots is saved. Overlays are created as the difference between the sets of snapshots, which accounts for a very small set of overlays. While the client has to send two overlays, there are fewer base VM dependencies. In addition, because the VM starts from a suspended state instead of from a stopped state, the VM Start time is faster.

KVM in NAT Mode and Port Redirection

One of the main complexities of the initial prototype was the KVM-to-Cloudlet-Host communication explained in Section 3 that allowed the mobile device to connect directly with the Face Recognition Server inside the VM.

The revised prototype starts the synthesized VM in NAT (Network Address Translation) mode instead of bridged mode so that the Guest VM can share the same IP address as the Cloudlet Host. However, in NAT mode the Guest VM is not directly accessible from the client. When the Cloudlet Server starts the synthesized VM in NAT mode, it includes the port that the Guest VM should listen on as a parameter. The Cloudlet Server maps an externally accessible port on the Cloudlet Host to the port assigned to the Guest VM. The externally accessible port number is sent to the client and the cloudlet host redirects all communication to the Guest VM. The tradeoff is that NAT is restricted to certain protocols (e.g., HTTP, SMTP, FTP) and cannot be bound to ports numbered lower than 1024 without root privileges.

Scalability could become an issue if the port request exceeds the number of available ports. However, our revision greatly simplifies deployment because the Windows service, Python runtime, CloudletIDPing Python script, and pyinotify are no longer necessary. In addition, NAT is more secure than bridged mode because the VM is firewalled from the outside.

5.2 Evaluation of Revised Prototype

We evaluated the revised prototype using the same applications as listed in Section 4.7. Application data is shown in Table 2. Overlay size is considerably smaller, except for FACE, due to application characteristics such as language, use of DLLs, configuration files, etc.

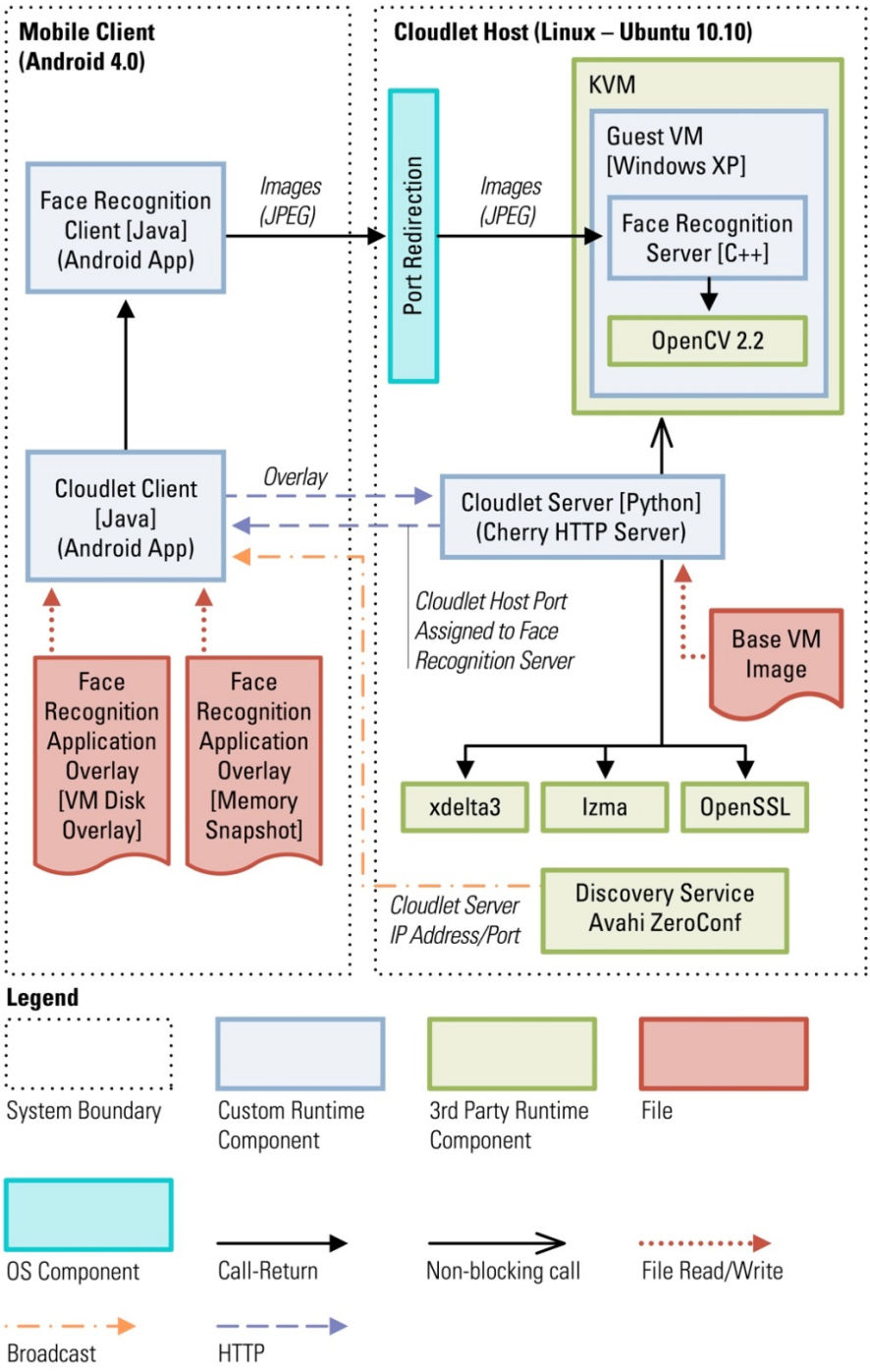


Fig. 7. Architecture for revised prototype

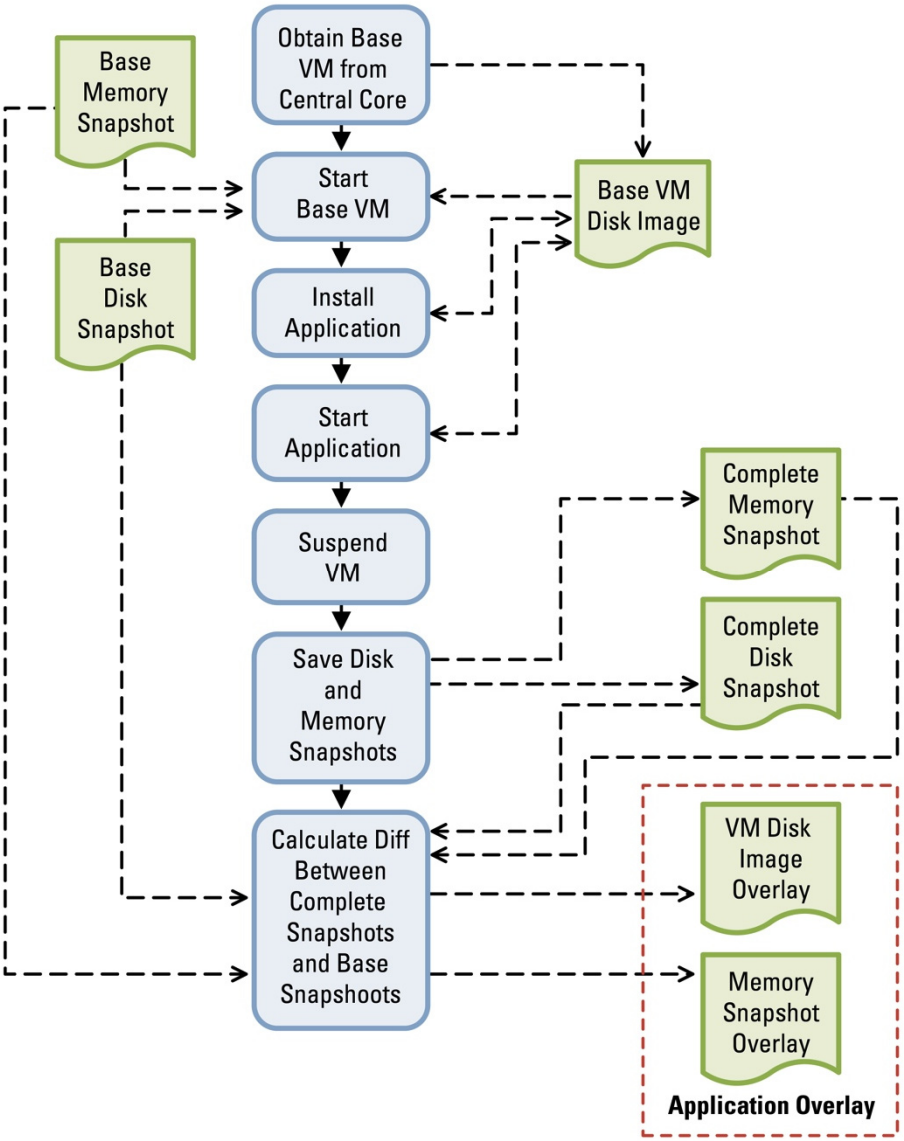


Fig. 8. Revised application overlay creation process

Table 2. Application Data for Revised Prototype

Application	Base VM Disk Image qcow2 (MB)	Base Disk Snapshot qcow2 (MB)	Base Memory Snapshot (MB)	Compressed VM Disk Image Overlay (MB)	Compressed Memory Snapshot Overlay (MB)
OBJECT	3558	17	554	94	293
FACE	2421	15	278	71	101
SPEECH	3558	17	554	86	257
NULL	3558	17	554	1	3

Average times for each step of the process and average energy consumption are shown in Figure 9.

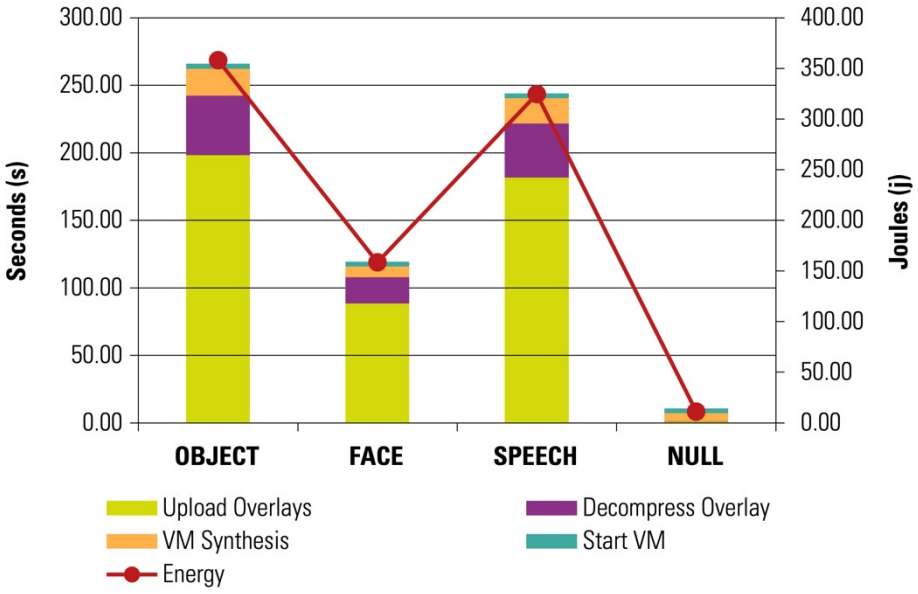


Fig. 9. Measures per application for revised prototype

Although the reduction in VM Synthesis and Start VM times are considerable with respect to the initial prototype, Upload Overlay and Decompress Overlay times are higher because of the total size of the combined overlays (disk plus memory). Consequently, energy consumption increased because of increased overlay size. However, implementation, application configuration, and VM-host-guest communication are simplified, and Start VM time is more consistent regardless of the operating system running inside the VM.

The bandwidth between the mobile device and the cloudlet during the experiments was approximately 13 Mbps, even when using 802.11n wireless. This lower-than-expected data rate may be caused by radio interference in the environment where the experiments were conducted. For the revised prototype to pay off, the efficiencies gained in VM Synthesis and Start VM would require supplementation with greater bandwidth.

6 Related Work

A considerable amount of work, conducted as early as 2001, relates to code offload from mobile devices to cloud environments [5][6][7][8][9][10][11][12][13][14][15][16][17][18][19][20]. However, this work in cyber-foraging from mobile devices assumes that acceptable networking conditions prevail between a mobile device and its offload site. Although the bandwidth and latency of this connectivity varied, a universal assumption in previous work was that connectivity was "good enough." To the best of our knowledge, our work is the first to investigate the challenges of cloud offload in hostile environments and to propose an architectural solution to the problem.

One example of closely related work is MAUI [25]. MAUI is a system that enables the fine-grained energy-aware offload of mobile code to offload elements, with minimum programmer effort—code annotations indicate methods that could be executed remotely. However, this approach is platform-specific (Microsoft .NET) and therefore would limit the applications that could be offloaded.

Another closely-related effort is CloneCloud [43]. Unlike with MAUI, applications do not require modification. This work also assumes connection to the cloud, but it could be implemented such that threads are migrated to a cloudlet instead of a cloud. However, supported platforms are limited (e.g., Java VM, Android Dalvik VM, Microsoft .NET) and deployment and hardware requirements would be difficult to achieve in some hostile environments.

Other work that establishes a foundation for cyber-foraging includes

- Goyal and Carter propose a VM-based approach in which a discoverable virtual machine server acts as a surrogate to run client application code [10]. This approach addresses security issues but requires the surrogate to be connected to the internet to locate and download client application code and also requires code to be partitioned at development time.
- Similar to this approach is the Locusts framework that enables the discovery of cyber-foraging resources (surrogate peers). Peers can offload coarse-grained tasks to other peers as well as process offloaded tasks from other peers [44]. Applications must be partitioned in advance into locally executed code and remotely executable tasks.
- Work proposed by Chen et al is not VM-based, but includes a mechanism for deciding whether to execute locally or remotely based on size of method input data and wireless channel conditions, and whether to interpret bytecode or compile native code [45]. The solution only works for Java code and applications must be modified to enable code offload.
- Kemp et al. propose an approach that leverages the Ibis high-performance distributed computing middleware [11]. The server portion is sent from the mobile device to the surrogate at runtime but applications must be written as distributed applications using the Ibis programming environment and the server.

Our work implements an instance of the cloudlet strategy presented by Satyanarayanan that uses a thick-client approach (native app) instead of a thin, VNC-based client [23]. The advantage of the VNC approach is that applications would not require any modification because they would run completely on the server. However, many applications use sensors to capture information about the environment. A better fit for our scenario involves splitting the application into a very simple native app that runs on the mobile device to capture manual or sensed input and a server portion that runs the expensive computation. The advantage of the VM-based approach is simplicity of setup and deployment that relies on generic servers running pre-configured Base VMs (cloudlets) and cloudlet-ready mobile devices loaded with overlays for computation-intensive applications. There is no need for special hardware or middleware.

7 Conclusions and Future Work

There is substantial consensus that cloud offload of resource-intensive application execution is a core technique in mobile computing. In this paper, we have described a reference architecture for code offload in hostile environments and presented two viable implementations along with architectural tradeoffs. A difficult problem exposed by this architecture involves rapid delivery of large application overlays to offload sites as well as rapid application ready time.

Current and future work includes 1) other forms of code offload, such as demand paging from the cloud, that consume less energy and provide better launch times, but require reliable communication between cloudlets and the cloud [22] 2) rapid VM synthesis, and 3) extension of the discovery protocol to enable VM caching so that overlays do not always have to be transmitted.

Acknowledgements. This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

References

1. Frenzel, L.: Street-Ready Smart Phone Enhances First Responder Communications. Electronic Communications (2012), http://electronicdesign.com/article/communications/streetready-smart-phone-enhances-responder-communications-73646
2. Kozlowski, J.: Smartphone and Tablet Apps for First Responders. EMS1.com (2012), http://www.ems1.com/ems-products/communications/articles/1129715-Smartphone-and-tablet-Apps-for-first-responders/
3. Morris, E: A New Approach for Handheld Devices in the Military. Software Engineering Institute Blog (2011), http://blog.sei.cmu.edu/post.cfm/a-new-approach-for-handheld-devices-in-the-military
4. Satyanarayanan, M.: Fundamental Challenges in Mobile Computing. In: Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing, pp. 1–7. ACM, New York (1996)

5. Balan, R., Flinn, J., Satyanarayanan, M., Sinnamohideen, S., Yang, H.: The Case for Cyber Foraging. In: Proceedings of the 10th ACM SIGOPS European Workshop, pp. 87–92. ACM, New York (2002)
6. Balan, R., Gergle, D., Satyanarayanan, M., Herbsleb, J.: Simplifying Cyber Foraging for Mobile Devices. In: Proceedings of the 5th International Conference on Mobile Systems Applications and Services, pp. 272–285. ACM, New York (2007)
7. De Lara, E., Wallach, D.S., Zwaenepoel, W.: Puppeteer: Component-based Adaptation for Mobile Computing. In: Proceedings of the 3rd Conference on USENIX Symposium on Internet Technologies and Systems, p. 14. USENIX Association, Berkely (2001)
8. Flinn, J., Narayanan, D., Satyanarayanan, M.: Self-Tuned Remote Execution for Pervasive Computing. In: Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems, pp. 61–66. IEEE, New York (2001)
9. Flinn, J., Park, S., Satyanarayanan, M.: Balancing Performance, Energy Conservation and Application Quality in Pervasive Computing. In: Proceedings of the 22nd International Conference on Distributed Computing Systems, pp. 217–226. IEEE, New York (2002)
10. Goyal, S., Carter, J.: A Lightweight Secure Cyber Foraging Infrastructure for Resource-constrained Devices. In: Proceedings of the 6th IEEE Workshop on Mobile Computing Systems and Applications, pp. 186–195. IEEE, Washington D.C (2004)
11. Kemp, R., Palmer, N., Kielmann, T., Seinstra, F., Drost, N., Maassen, J., Bal, H.: eyeDentify: Multimedia Cyber Foraging from a Smartphone. In: Proceedings of the 11th IEEE International Symposium on Multimedia, pp. 392–399. IEEE, New York (2009)
12. Ok, M., Seo, J.-W., Park, M.-S.: A Distributed Resource Furnishing to Offload Resource-Constrained Devices in Cyber Foraging Toward Pervasive Computing. In: Enokido, T., Barolli, L., Takizawa, M. (eds.) NBiS 2007. LNCS, vol. 4658, pp. 416–425. Springer, Heidelberg (2007)
13. Satyanarayanan, M.: Pervasive Computing: Vision and Challenges. IEEE Personal Communications, 10–17 (2001)
14. Christensen, J.: Using RESTful Web Services and Cloud Computing to Create Next-Generation Mobile Applications. In: Proceedings of the 24th ACM SIGPLAN Conference Companion on Object-Oriented Programming Systems Languages and Applications, OOPSLA 2009, pp. 765–766. ACM, New York (2009)
15. de Leusse, P., Periorellis, P., Watson, P., Maierhofer, A.: Secure and Rapid Composition of Infrastructure Services in the Cloud. In: Proceedings of the 2nd International Conference on Sensor Technologies and Applications, pp. 770–775. IEEE, New York (2008)
16. Kumar, K., Lu, Y.: Cloud Computing for Mobile Users: Can Offloading Computation Save Energy? IEEE Computer 43, 51–56 (2010)
17. Li, X., Zhang, H., Zhang, Y.: Deploying Mobile Computation in Cloud Service. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) Cloud Computing. LNCS, vol. 5931, pp. 301–311. Springer, Heidelberg (2009)
18. Marinelli, E.: Hyrax: Cloud Computing on Mobile Devices using MapReduce. Technical Report, Carnegie Mellon University (2009)
19. Palmer, N., Kemp, R., Kielmann, T., Bal, H.: Ibis for Mobility: Solving Challenges of Mobile Computing Using Grid Techniques. In: Proceedings of the 10th Workshop on Mobile Computing Systems and Applications, article No. 17. ACM, New York (2009)
20. Zhang, X., Schiffman, J., Gibbs, S., Kunjithapatham, A., Jeong, S.: Securing Elastic Applications on Mobile Devices for Cloud Computing. In: Proceedings of the 2009 ACM Workshop on Cloud Computing Security, pp. 127–134. ACM, New York (2009)
21. Dilger, D.: First Look: Using iPhone 4S with Siri Voice Assistant (with Videos). Apple Insider (2011), http://www.appleinsider.com/articles/11/10/14/first_look_using_iphone_4s_with_siri_voice_assistant.html

22. Ha, K., Lewis, G., Simanta, S., Satyanarayanan, M.: Code Offload in Hostile Environments. Technical Report, Carnegie Mellon University (2011)
23. Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: The Case for VM-Based Cloudlets in Mobile Computing. IEEE CS Pervasive Computing, 14–23 (2009)
24. Lehr, W., McKnight, L.: Wireless Internet Access: 3G vs. WiFi? Center for eBusiness @ MIT (2002)
25. Cuervo, E., Balasubramanian, A., Cho, D., Wolman, A., Saroiu, S., Chandra, R., Bahl, P.: MAUI: Making Smartphones Last Longer with Code Offload. In: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, pp. 49–62. ACM, New York (2010)
26. Wolbach, A.: Improving the Deployability of Diamond. Technical Report, Carnegie Mellon University (2008)
27. Richardson, T., Stafford-Fraser, Q., Wood, K.R., Hopper, A.: Virtual Network Computing. IEEE Internet Computing 2(1), 33–38 (1998)
28. Microsoft Corporation: TCP/IP and NBT configuration parameters for Windows XP, http://support.microsoft.com/kb/314053
29. xdelta.org, http://www.xdelta.org
30. 7-Zip.org: LZMA SDK (Software Development Kit), http://www.7-zip.org/sdk.html
31. OpenSSL.org: OpenSSL: The Open Source Toolkit for SSL/TLS, http://www.openssl.org/
32. KVM: Kernel-Based Virtual Machine, http://www.linux-kvm.org/
33. KVM: Guest Support Status, http://www.linux-kvm.org/page/Guest_Support_Status
34. CherryPy: Cherry Py – A Minimalist Python Web Framework, http://www.cherrypy.org/
35. Ubuntu: KVM/Networking – Community Ubuntu Documentation, https://help.ubuntu.com/community/KVM/Networking
36. Github: seb-m/pyinotify—Github, https://github.com/seb-m/pyinotify
37. Avahi.org: Avahi, http://avahi.org/
38. Zeroconf: Zero Configuration Networking (Zeroconf), http://www.zeroconf.org
39. OpenCV: Welcome – Open CV Wiki, http://opencv.willowgarage.com/wiki/
40. MOPED: MOPED: Object Recognition and Pose Estimation for Manipulation, http://personalrobotics.ri.cmu.edu/projects/moped.php
41. SPHINX-4. Sphinx-4 – A Speech Recognizer Written Entirely in the Java Programming Language, http://cmusphinx.sourceforge.net/sphinx4/
42. Monsoon Solutions: Power Monitor, http://www.msoon.com/LabEquipment/PowerMonitor/
43. Chun, B., Ihm, S., Maniatis, P., Naik, M., Patti, A.: CloneCloud: Elastic Execution between Mobile Device and Cloud. In: Proceedings of the 6th European Conference on Computer Systems, pp. 301–314. ACM, New York (2011)
44. Kristensen, M.D.: Execution Plans for Cyber Foraging. In: Proceedings of the 1st Workshop on Mobile Middleware, article no. 2. ACM, New York (2008)
45. Chen, G., Kang, B., Kandemir, M., Vijaykrishnan, N., Irwin, M., Chandramouli, R.: Studying Energy Trade Offs in Offloading Computation/Compilation in Java-Enabled Mobile Devices. IEEE Transactions on Parallel and Distributed Systems 15(9), 795–809 (2004)

Nihao: A Predictive Smartphone Application Launcher

Chunhui Zhang, Xiang Ding, Guanling Chen, Ke Huang,
Xiaoxiao Ma, and Bo Yan

Computer Science Department, University of Massachusetts Lowell,
1 University Avenue, Lowell, Massachusetts, USA, 01854
{czhang,xding,glchen,khuang,xma,byan}@cs.uml.edu

Abstract. Increasingly large number of the applications installed on smartphones tends to harm the application lookup efficiency. In this paper, we introduce Nihao, a personalized intelligent app launcher system, which could help the users to find apps quickly. Nihao predicts which app the user will likely open next based on a Bayesian Network model leveraging the contextual information such as the time of day, the day of week, the user's location and the last used app with the hypothesis that the users' app usage pattern is context dependent. Through the field study with seven users over six weeks, we first validate the above hypothesis by comparing the prediction accuracy of Nihao with other predictors. We found that the larger UI change did not necessarily yield longer app lookup time as the app lookup time highly depended on the app icon position on screen, which suggested the prediction accuracy was the most important factor in designing such a system. At the end of the study, we conducted a user survey to evaluate Nihao qualitatively. The survey results show that five out of seven users were quite satisfied with the prediction of Nihao and thought it could help to save both app lookup and management time by ranking the app icons automatically while Nihao did not help the other two users much since they used their phones primarily for calling and texting (not for apps).

Keywords: mobile applications, Android applications, application usage prediction, context awareness, adaptive UI.

1 Introduction

As of March 2012, the number of apps and games in Android market reached about 620,000 and the total number of downloads was estimated to be over nine billion [1]. Increasingly large number of the apps installed on smartphones tends to decrease the app lookup efficiency and more time is required to manage them. A typical Android phone launcher by default places all app icons in the *app drawer* and the apps are usually ordered according to their installation time or alphabetically. Furthermore, the users could create shortcut for their favorite apps on the homescreens directly or manage the shortcuts in folders. With a large number of apps installed, the users usually launch an app either by looking

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 294–313, 2013.

up the shortcut screen by screen (Android phone usually has more than one homescreens) or by going through the prohibitively long app list in the *app drawer*. In addition, the task of customizing and arranging the app shortcuts consumes considerable amount of time and effort.

To shorten the app lookup and management time, we developed Nihao, a personalized intelligent app launcher system, which could automatically rank the apps by predicting how likely they tend to be opened. The prediction is based on a Bayesian Network model leveraging various contextual information such as the time of day, the day of week, the user's location and the last used app, with hypothesis that the users' app usage pattern depends on those context. For instance, someone likes to read *Technology Review* in the morning and use *Gtalk* to contact his wife at the end of the day during weekdays while he likes to play games using his phone during weekends. Another example is that someone is likely to use *Yelp* [2] to read reviews for dishes when she is in a restaurant while she tends to use *Google Map* to find interesting places in the destination when waiting for the plane in the airport. Similarly for the context of the app correlation, the users may use *eBay* app after using *Amazon* app.

When designing such a system involving adaptive user interface, whether the UI change will hurt the usability is always a concern [3]. Some researchers believe that the users need a predictable UI while others argue that the accurate prediction could mitigate the confusion caused by the UI change. In this paper, we aim to find out the feasibility and efficiency of the dynamic app launcher UI based on the predictions.

With the aforementioned expectation, hypothesis and concern in mind, we conduct a field study with seven users for six weeks. As the data analysis results show, the performance of Nihao in term of app lookup time outperformed the default launcher when considering only downloaded apps. (Built-in apps such as *phone*, *text messaging*, *browser* are usually opened through the *quick launch bar* at the bottom of the homescreen in a Android phone and require little lookup time). Then, the hypothesis was proved by demonstrating that Nihao predicted more accurately than the benchmark predictor (app usage frequency based predictor) and other predictors. Finally, we found that the larger UI change did not necessarily yield longer app lookup time and the app lookup time highly depended on the app icon position on screen, which suggested the prediction accuracy was the most important factor in designing such a system.

At the end of the study, we conduct a user survey to evaluate Nihao qualitatively. The survey result shows that five out of seven users were quite satisfied with the prediction of Nihao and thought it could help to save both app lookup and management time by ranking the app icons automatically while Nihao did not help the other two users much since they used their phones primarily for calling and texting.

The contributions of this work include: 1) a new and effective UI design for intelligent smartphone app launcher, 2) an effective app usage prediction method based on a probabilistic model leveraging contextual information such as time,

location and app correlation, 3) a reference system design and implementation of such a context-ware personal assistant to speed up app lookup.

The rest of this paper is organized as follows. In Section 2, we compile the related works. Section 3 discusses the interface design and the prediction model. Section 4 presents details of the system design and implementation. Next, we show the system evaluation results in Sections 5. Finally, Section 6 concludes with planned future work.

2 Related Work

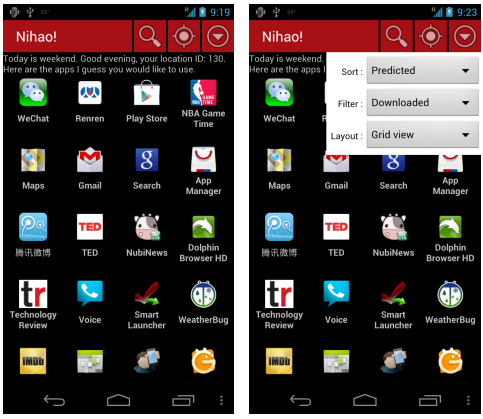
Several recent works have explored the smartphone usage and its correlation to contexts. H. Falaki et al. suggest that smart phone service should be customized due to the vast usage diversity among users [4]. M. Bohmer et al. find that app usage is correlated to contexts such as time of the day and the user's location using the dataset collected through their app recommendation tool *appazzar* in Android market [5] . T.M.T.Do et al. show two dependencies of app usage, i.e. place and social context based on data collected from Nokia phones [6]. Most recently, Ke Huang et al. proposed to use several contexts to predict app usage and found that the app correlation is the most influential factor [7]. So far, these mentioned researches are all based on the data collected from the smart phones. In contrast, Q. Xu et al. reveal day-time app usage pattern and show the app correlation through analysis based on a Tier1 ISP's dataset [8]. Instead of focusing on the context-based data analysis, our work emphases on the design of a complete system and the analysis of its effectiveness.

There is a rich body of literature regarding the adaptive user interface design. About twenty years ago, Andrew and Ben studied the *split menus* which is a menu comprises both a static and an adaptive part with a line separating them [9]. The adaptation was based on the selection frequency which could directly reflect the user's preference. Leah and Joanna showed that adaptive user interface is more beneficial in a small size screen than in a large size screen [10], which strengthens our confidence in designing Nihao. Melanie Hartmann summarized the challenges in designing adaptive user interface in his survey paper [3]. Recently, Santi. P. et al. have organized the outgoing call prediction result in form of *Intelligent Address Book* where the predicted callee IDs are ranked based on a computed score [11], while our system foresees the apps that are likely to be opened and organizes the app icons in a grid or list view.

Recently, several research groups have been focusing on the app recommendation systems for smart phone users. Bo Yan et al. developed a collaborative app recommendation system (*AppJoy*) considering the app usage history such as *frequency*, *duration* and *recency* [12]. M. Bohmer et al. present *Appazaar*, which is a context-aware recommender system [13]. In contrast to those works, our system focuses on predicting which already installed apps to be opened next.

3 Approach

If we had a good app launch prediction system, there are at least two usage scenarios to leverage this kind of app prediction. One use case is to rank the apps on launch pad based on the scores derived from the prediction model (i.e. Figure 1a). This use case falls into the category of intelligent user interface design and focuses on fast app lookup. Another use case is to pre-load apps that are likely to be launched into memory based on the prediction model [14]. These two use cases are different as the later focuses on launching the app quickly after the user finds the app, while the first one helps the user finds the app quickly. In this paper, we focus on the first use case, aiming to build an intelligent UI to assist app lookup. Thus, we propose a context-based app ranking method by leveraging the context such as *TimeOfDay*, *DayOfWeek*, *Location* and *LastUsedApp*.



(a) Screenshot1 (b) Screenshot2

Fig. 1. Nihao screenshots

3.1 Contextual Information

Based on the extensive data analysis result reported recently (see the related work), many contextual information have been found correlated to the mobile app usage in certain ways. Those contexts include time, the user's location, the correlation between apps and the social context etc. We believe that the app usage pattern is much more complex and the user's app usage behavior is influenced by much richer context. By balancing the importance of the context and the engineering effort, we chose the four aforementioned contexts.

First, *TimeOfDay* is defined as {morning(5am-11am): 0; noon(11am-2pm): 1; afternoon(2pm-6pm): 2; dinner(6pm-9pm): 3; evening(9pm-5am): 4}. Actually, we originally proposed a variable *HourOfDay* which has 24 values. But we

changed it to the less fine-grain variable *TimeOfDay* because it required shorter learning period and much less space for storing the historical data which is a concern due to the limited disk storage in a smartphone. Then, we introduced a variable *DayOfWeek*. For the same reason, we defined it as a binary variable with 0 representing the weekdays and 1 representing the weekends. Next, *Location* is represented by location ID computed through our significant place algorithm that is further explained in the next section. Finally, the correlation between apps is considered. For the sake of simplicity, an app is treated only correlated with the last used app (within a user interaction *session*, which is defined as the time between the screen is turned on and off) which follows the order-one Markov model. The value of the variable *LastUsedApp* is simply the string of the app name.

3.2 Nihao Model

Based on the variables introduced, we construct a Bayesian Network for Nihao (Figure 2) assuming that the app usage is dependent on the time of day, day of week, the user's location and which app was used before, where location is dependent on both temporal variables. Every time the user opens Nihao, each app is assigned a score calculated as the conditional probability of the app to be opened according to the following equation,

$$Score(A) = P(A|D, L, T, A') = \frac{P(A|D, L, T) * P(A|A')}{P(A)} \tag{1}$$

and then the app icons are arranged in a grid view (optionally a list view) from left to right and from top to bottom. When we actually calculate the scores, the posterior probability of an app, P(A), is assumed to be equal for all apps.

All the probabilities were computed against the past three weeks' historical data to capture the most recent app usage behavior. However, what is the optimal value for this window size is still open and we will try to make this value personalized and adaptive in the future.

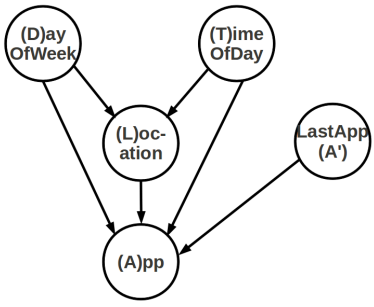


Fig. 2. Bayesian network model for Nihao

4 System Design and Implementation

Nihao follows the client-server model. The client runs on a smartphone and is responsible for providing the user interface, recording the app usage and the associated contextual data. The server runs on remote PCs for storing the historical data and conducting machine learning tasks such as learning new models. But, please note that the client could work when the server is down since all the UI required statistics are cached locally. For the communication between the client and the server, a customized protocol generated through Google ProtoBuf [15] over http is used. Details are discussed in the following subsections.

4.1 Client

The client software contains several components including the UI, a model manager, a local database and a background service (Figure 3).

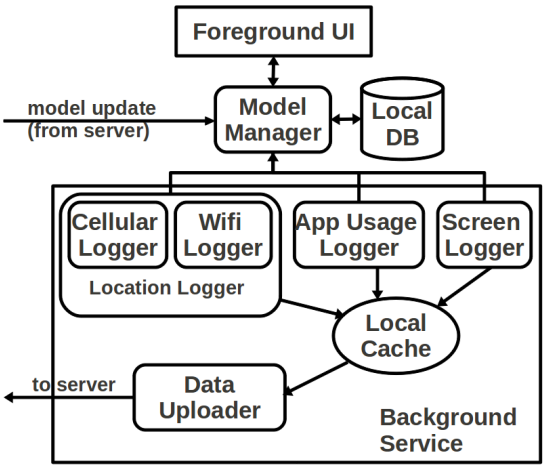


Fig. 3. Client software architecture

First, the major task of the UI is to present the app icons. Following the tradition of Android phone, Nihao uses the grid view as the default configuration and provides the list view as an option (Figure 1a). Users may like to switch between the grid view and the list view. Considering the code clarity and efficiency we put both views in the same activity (term in Android) and play with the view's visibility instead of assigning views to different activities. Furthermore, Nihao allows users to use popular static ranking methods i.e. *alphabetical* and *new app first* ranking. For both methods, installed apps will be showed thus it is particularly helpful when Nihao has not learned much at the beginning. In addition, the search function is also implemented within Nihao. However we found that it was seldom used by the subjects during the study which suggests that people prefer

to open apps by browsing instead of searching. Another function provided by Nihao UI is the place check-in. As a location-ware system, this function allows Nihao to learn place semantics opportunistically.

Second, as the name suggested, the model manager prepares the statistics required by the model in local DB and computes the score for the apps. For example, the model manager implements the significant place recognition algorithm to produce the location ID. According to the Bayesian Network model used, the statistics are organized in three local DB tables i.e. *location*, *context* and *correlation*. Each record in *location* table represents a location with a unique ID. *context* table maintains the statistics to calculate $P(A|D, L, T)$ and *correlation* table stores the data to compute $P(A|A')$.

Finally, let us discuss the background service. As the most important component of Nihao, it further comprises several objects such as the cellular logger, the WiFi logger, the app usage logger, the screen logger, the data uploader and a local cache. The cellular logger and the WiFi logger is used together to provide the input of the significant place recognition algorithm such as the cell ID, the connected WiFi AP MAC address and the list of nearby WiFi AP MAC addresses. The app usage logger is used to monitor the app usage i.e. listen on the app open event. However, Android does not directly provide event-based APIs for such a task. Fortunately, the APIs to find the top activity, the activity the user is currently interacting with, is offered. So Nihao eventually generates app open events itself by polling the top activity for every one second. The screen logger is responsible for recording the screen on/off events which indicates the start and the end of the phone usage *sessions*. Depends on whether it is the first app used in a *session*, Nihao computes the scores for apps differently. App ranking is only based on $P(A|D, L, T)$ if no app is used so far within the session while $P(A|D, L, T) * P(A|A')$ is calculated if some apps has been opened. The data uploader is responsible for uploading the data generated by the loggers to the server. The uploading process is totally independent from the UI thread since the data needed for UI are stored in local DB in real time and at the same time the data are put into the local cache. To save the battery and reduce the 3G/4G network traffic, the data are uploaded when the WiFi network is available and several *records* are batched into a single HTTP POST when uploading. In addition, considering the user privacy, all the sensitive data such as Android ID, Wifi Mac address etc, are hashed.

4.2 Server

It is valid to ask why Nihao needs the server since the model required statistics are stored locally on the client and the client could work even when the server is down. Simply put, the server is used to preserve the app usage history for users and run the complex machine learning algorithms. Even though Nihao currently uses the fixed model for all users, more suitable model may exist for different users. To find that out, based on the historical raw data on the server, we could keep experimenting with the new models. When a better model is found, Nihao could push the updated code and the statistics needed for the new model to the clients.

Nihao server is implemented based on Play Framework [16] and MySQL database. Instead of storing the data received in database directly, Nihao server stores the data in daily files first. It is more efficient to do so thus the scalability of the server is increased. In addition, for the purpose of storing the historical raw data, the daily files are easier and more convenient to store and be moved around. Then, a scheduled job is designed to read the data from the files and store them to the database every 2:00AM in the morning. It is better to run various machine learning algorithms against database since search is often involved and it could benefit from the record indexing in the database.

4.3 Significant Place

Our significant place recognition method involves two steps. First, an automatic algorithm is used to compute the place ID (or the location ID, we use them interchangeably throughout the paper). Second, instead of guessing the semantic meaning of the place, the *check-in* function is implemented as the part of the UI to allow the user to teach the system opportunistically. But be noted that Nihao model is based on the place ID directly instead of the semantic label at this time.

Recognition Algorithm. Nihao jointly uses the identifiers of cellular network and WiFi network to recognize the significant places by assuming that the most in-door significant places have WiFi network. Cellular network is mainly used to recognize outdoor places. In Nihao, a significant place is represented by a tuple: (a set of cell IDs, a set of connected WiFi AP MAC addresses, a set of nearby WiFi AP MAC addresses). The first element in the tuple is a set of cell IDs instead of a single cell ID because a significant place may reside on the boundary of the adjacent cells in the cellular network. For the similar reason, the second element is also defined to be a set. The third element could be acquired through the WiFi scan and it is naturally a set.

The recognition process follows a decision tree (Figure 4). First of all, the algorithm checks whether the WiFi network is connected. If it is connected, it then tries to match a tuple in the local database and return the place ID. Here, by match, we mean the current connected WiFi MAC address is contained in the set of connected WiFi AP MAC addresses in an existing tuple. If the WiFi network is not connected, the algorithm will check whether the WiFi scan result (a set of nearby AP MAC addresses) presents. The scan is conducted by the WiFi logger and multiple consecutive scans are issued in the same scanning session to find as much APs as possible. If the WiFi scan result is none-empty, the algorithm will again try to compare the current WiFi scan result(A) to the the third variable of the tuples in DB (B). An existing place ID will be returned and the WiFi scan result will be merged into the nearby APs set in the tuple if there are at least two overlapped WiFi APs in (A) and (B). Otherwise, a new record will be created and a new place ID will be returned. Finally, the cell ID is used to match the record in a similar way if the WiFi scan result is empty.

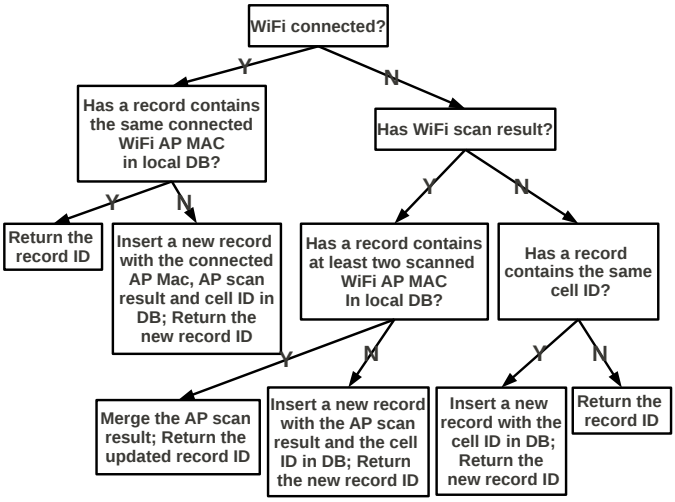


Fig. 4. Decision tree for significant place recognition

After the six weeks' filed study, the number of recognized places ranges between 4 and 60 for different users and the largest WiFi AP cluster contains 483 APs.

Place Check-In. To acquire the location semantics, we design the *place check-in* function and the places include home, friend's home, work place, school, restaurant, shopping, gym, theater, outdoor and transportation related places. When the user checks in a place, the semantic label is assigned to the current place ID computed by the recognition algorithm and the place records with the same label will be merged. Basically, the place check-in function is used to group the places by their semantic meaning because for instance Nihao should predict in the same way when a user in different restaurants no matter where they are geographically.

4.4 Data Format and Protocol

Nihao uses Google ProtoBuf to manage the data format used to store and upload the data. Listing 1 shows the .proto file which is used to define the structure of the data and could be compiled to generate the code for serializing the data.

Listing 1. Nihao .proto file (1)

```
message Record {
  enum RecordType {
        APPUSAGE = 1;
        SCREEN = 2;
        WIFI = 3;
```

```
        CELLULAR = 4;
    }
    message AppUsage {
        required string package_name = 1;
    }
    message Screen {
        required bool on_off = 1;
    }
    message WiFi {
    ...
    }
    message Cellular {
    ...
    }
    required int64 timestamp = 1;
    required RecordType type = 2;
    optional AppUsage app_usage = 3;
    optional Screen screen = 4;
    optional WiFi wifi = 5;
    optional Cellular cellular = 6;
}
```

According to the message *Record*, four RecordTypes and their structure are defined and the corespondent variables are designed to be optional. Thus, four different loggers could share the same .proto file by using only the variable interests them. *timestamp* and *type* are two required variables since for each record Nihao needs to know when it is generated and what the type is.

The Nihao client uploads the data through Http Post with the body being the data serialized by ProtoBuf. To increase the data efficiency, we define message *RecordPool* (Listing 2) to upload the data in batch mode.

Listing 2. Nihao .proto file (2)

```
message RecordPool {
    required int32 version_number = 1;
    required string android_id_hash = 2;
    required string email_hash = 3;
    repeated Record records = 5;
}
```

5 Evaluation

We conducted a field study for six weeks with seven subjects, who are college students and professionals. At the beginning of the study, we installed Nihao on the participants' Android phones. The first three weeks were the learning period for Nihao, during which we asked the subjects to keep using their default device

UI to launch apps while Nihao was running in the background, collecting app usage data and related contextual data. In the following three weeks, we asked the subjects to use Nihao to launch apps as much as possible.

5.1 App Usage

Figure 5 and Figure 6 show the app usage statistics Nihao collected and demonstrate the usage diversity of the subjects. In both figures, for each user we separately show the usage of downloaded apps and built-in apps (e.g. phone, messaging, browser etc.) since we wanted to understand how much the downloaded apps, as the primary target of Nihao, were used. As Figure 5 shows, User1 used

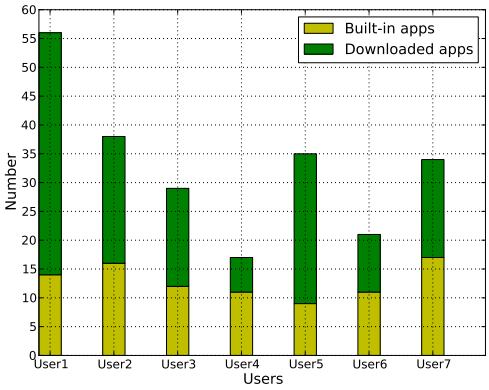


Fig. 5. Number of used apps for seven users

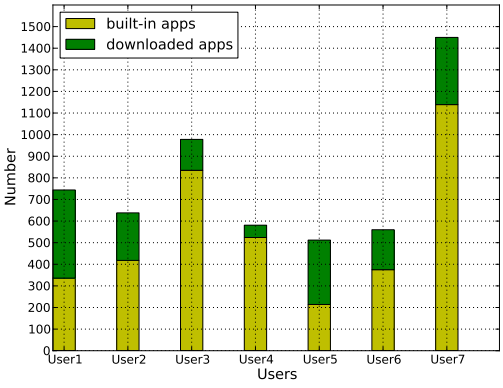


Fig. 6. App launch count for seven users

56 apps (the most) during the test period while User4 just used 17 apps (the least). Users used 33 apps on average. Considering the ratio of the downloaded apps to the built-in apps, User1 is 3.0 (highest) while User4 is 0.55 (lowest), which indicates that User1 had much broader app usage behavior than User4. In Figure 6, User7 launched apps for more than 1400 times among which about 3/4 times were for built-in apps. For User5, although the total launches were much less than that of User7, he mostly used downloaded apps.

5.2 Launched App Position

First, we evaluated the model accuracy of Nihao using the metric *launched app position*, which is the icon position in the grid view when the user clicks it. The position starts with 1 and increases from left to right and top to bottom fashion. Currently we show 4 app icons in each row of the grid view. We compared Nihao with other predictors, such as frequency based predictor, app correlation based predictor, time based predictor and location based predictor. For Nihao, *launched app position* was recorded directly through Nihao UI. For other predictors, this metric was calculated based on the dataset collected through the six weeks user study. Simulating the real usage scenario, we used first three week's data as training dataset to predict the next three weeks' app launch. In addition, we progressively added the previous app launch records to the training dataset when predicting the next one.

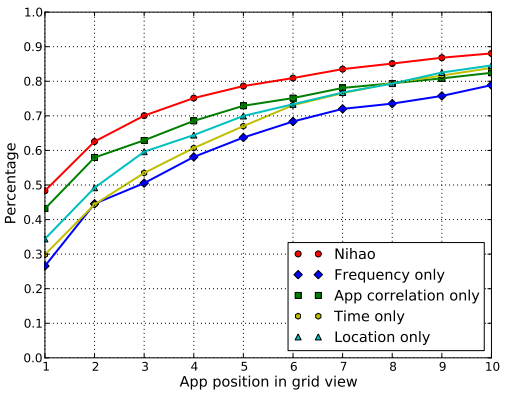


Fig. 7. CDF of launched app position (seven users combined)

As Figure 7 shows, 76% of the app launches were within first four positions (first row in the grid view) for Nihao while the number is only 58% for the frequency based predictor. Other predictors' performance were in between.

This result validates our hypothesis that app usage is correlated with the contextual information such as time, location and last used apps and a context-based predictor can be effective. However, Nihao model is not as accurate across all users. For User1, Nihao is the most accurate predictor (Figure 8), but for User3, Nihao does not obviously predict better than other predictors (Figure 9). The reason could be found by analyzing the most used apps of individual users. Take User3 for example, top five most used apps took up 81% of the total app launches (Figure 11) while the number was just 50% for User1 (Figure 10). In other words, User3 used top five apps for the most of times, thus the frequency-based predictor worked fairly good and that user can hardly take advantage of Nihao prediction. On the other hand, User1 had much broader app usage pattern and he benefited a lot from Nihao model. As a future work, we plan to develop a customized prediction model that uses different predictors depending on the user's app usage patterns.

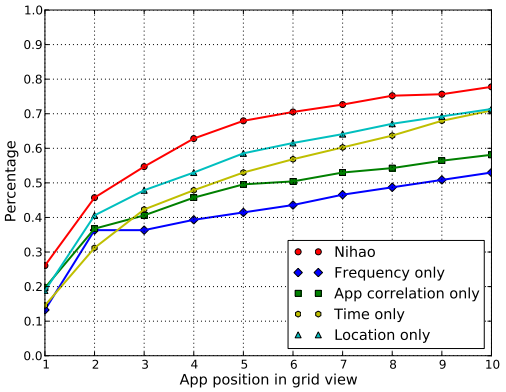


Fig. 8. CDF of launched app position for individual users (User1)

5.3 App Lookup Time

Next, we show the effectiveness of Nihao by analyzing how quick it could help its users to locate apps. We compare app lookup time using Nihao with the lookup time using the default device UI. We define the former as the time between when Nihao UI was showed and the first app was launched, and define the later as the time between when home screen was showed and the first app was launched. The default UI app lookup time is further separated into two variables, the lookup time for built-in apps and the lookup time for downloaded apps. As Figure 12 illustrated, the built-in apps in general had the shortest lookup time since apps like phone, messaging and browser were usually launched through quick launch bar at the bottom of the home screen, thus requiring little lookup time. But our focus is to compare the lookup time through Nihao with the time needed to launch downloaded apps through the default UI. In both cases, for 35% of

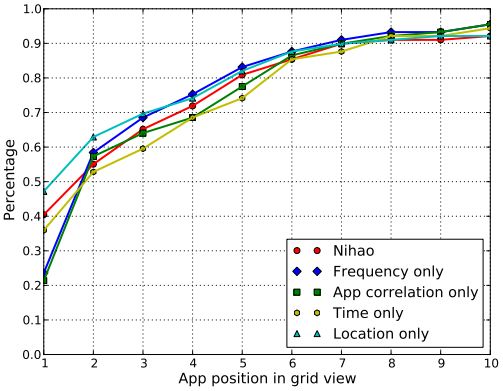


Fig. 9. CDF of launched app position for individual users (User3)

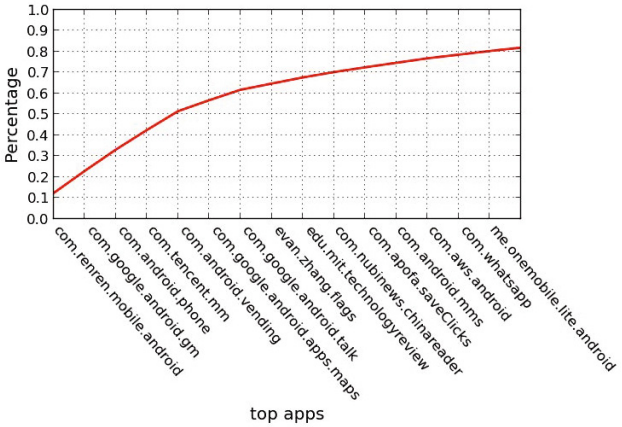


Fig. 10. Top apps launch count for individual users (User1)

times the apps were launched within 2.5 seconds. However, for about 70% of times, apps launched via Nihao required at most five seconds while only 60% of times required at most five seconds to launch a downloaded app. On average, lookup time for Nihao is 4.94s while the time for finding the downloaded apps through default launcher UI is 5.51s. Comparing to the default UI with high user familiarity, even the benefit in terms of app lookup time is not significant in general, (but it could be very significant for some users as Figure 13 shows), it is already a great achievement for Nihao to outperform within such a short period of learning time (three weeks). In addition, Nihao could potentially help save time required to manage apps (i.e. arranging app icons on home screen and assigning apps to different folders).

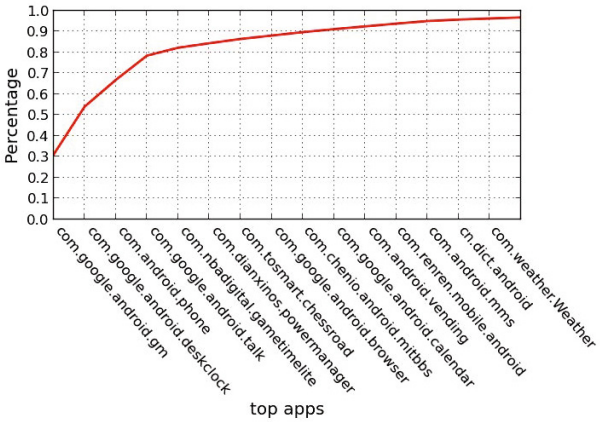


Fig. 11. Top apps launch count for individual users (User3)

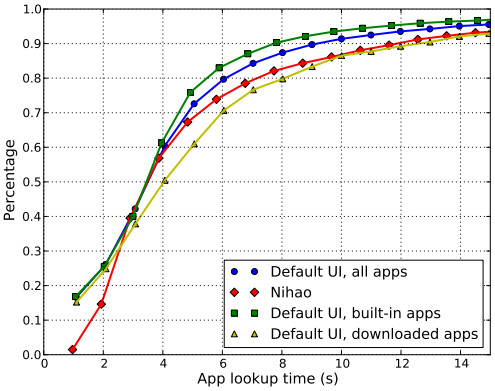


Fig. 12. CDF of app lookup time

5.4 Correlation between Launched App Position and App Lookup Time

After analyzing the *launched app position* and the *app lookup time*, it is interesting to see the correlation between them. In Figure 15, the x-axis represents the row number in a grid view and the y-axis represents the average time for finding the apps in a particular row with an error bar showing the standard deviation. As expected, the app lookup time is monotonically increasing as the row number in grid view increases which suggests that the more accurate prediction yields shorter app lookup time and it also proves that our eyes usually scan for apps row by row from the top. To further understand the correlation between the individual app position and the app lookup time, we show Figure 16 with the

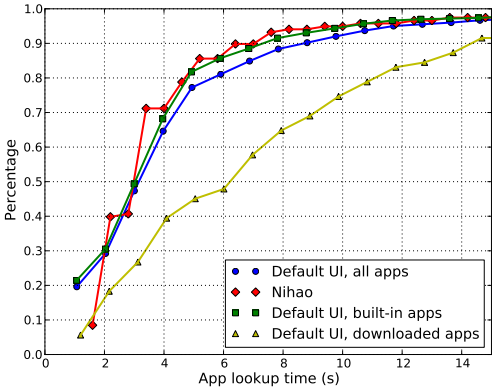


Fig. 13. App lookup time for individual users (User3)

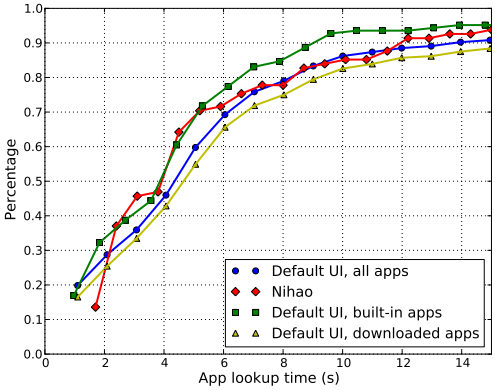


Fig. 14. App lookup time for individual users (User5)

x-axis representing the app icon position number in a grid view (This number increases with the position from left to right and from top to bottom). and the y-axis representing the average time for finding the app in that position. Unfortunately, we failed to find similar pattern as Figure 15. This result suggests that the users may start scanning the apps from left, right or even in the middle.

5.5 UI Predictability

In order to understand how dynamic/predictable the Nihao UI is (we assume that more dynamic UI yields less predictable UI), we introduce a metric *UI difference* between Nihao launches that is calculated as the number of positions on which the app changed between Nihao launches. For example, the difference

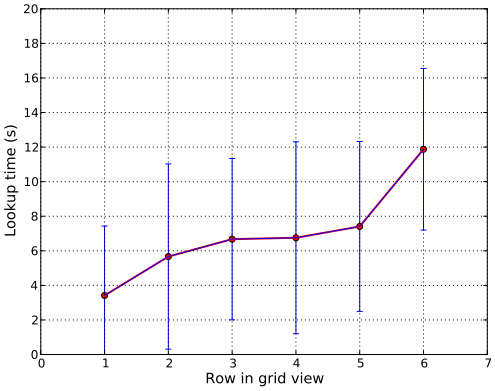


Fig. 15. Correlation between launched app position(in term of row in grid view) and lookup time

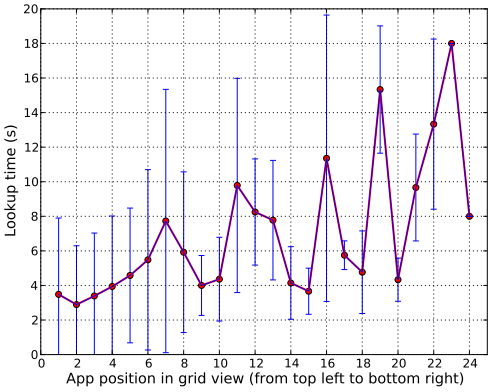


Fig. 16. Correlation between launched app position and lookup time

between (A B C) and (A C B) is two. Then, we use this metric to compare Nihao with other predictors (same as the predictors being compared with when we analyze the *launched app position*). As Figure 17 shows, for about 80% of the times, the number of the changed apps was at most five for the frequency based predictor, while for less than 30% of the times Nihao was as dynamic. In other words, Nihao UI was significantly more dynamic than the UI of the frequency based predictor. The app correlation only based predictor produced quite dynamic UI and the curve was fairly close to the curve of Nihao which means that the app correlation is the primary contributor to the highly dynamic UI of Nihao. The time only based predictor and the location only based predictor produce much less dynamic UI than Nihao, though its UI was still more dynamic than the frequency based predictor.

To understand whether the larger UI difference will produce longer app lookup time, we plotted Figure 18 with x-axis representing the number of the UI difference between the Nihao launches and y-axis representing the average app lookup time given the number of UI difference with the error bar showing the standard deviation. The figure shows that the larger UI difference does not necessarily require longer app lookup time. At the same time, we already know that the app lookup time is highly correlated to the app position (Figure 15). Thus, it allows us to conclude that the prediction accuracy is the most important factor than the UI predictability in designing such a system.

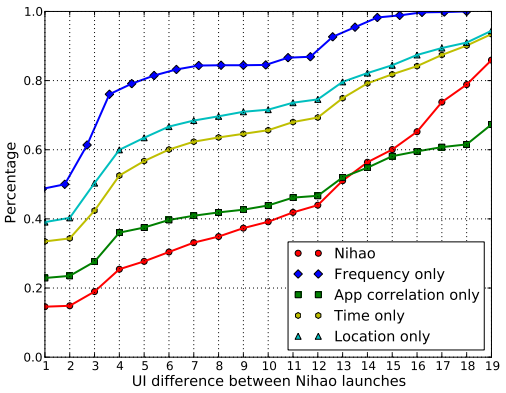


Fig. 17. CDF of UI difference between Nihao launches

5.6 User Survey

Finally, we conducted a user survey to better understand Nihao qualitatively. We first asked questions about how the users were satisfied with the predicted apps. Given the scale of one to seven, the average score is 6.3 which indicates that the users were quite satisfied. Furthermore, five out of seven users would like to keep using Nihao regularly after the test since they had become familiar with Nihao and chose it as their personal app management assistant. The other two users mainly used their phones for calling and texting thus Nihao was not as helpful to them. Next, we obtained user opinions on Nihao's UI design. First, six out of seven users preferred Nihao to be configured as a widget in one of the homescreens instead of a standalone app since one more click could be saved in the form of a widget. Second, all users preferred the grid view to the list view. Finally, by considering two typical usage scenarios for Nihao: 1) to find a specific app in mind quickly, 2) to browse and open some interesting apps to kill time), five out of seven users chose the first option which suggested that Nihao was largely treated as a serious app lookup tool instead of a time killer.

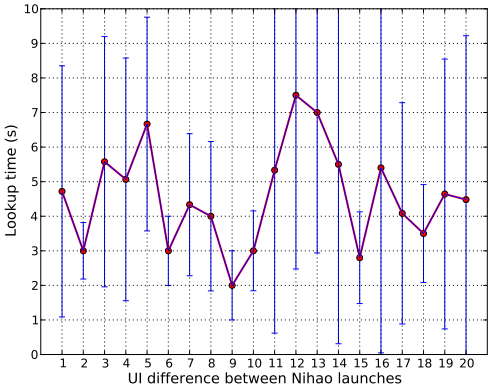


Fig. 18. Correlation between UI difference and app lookup time

6 Conclusion and the Future Work

To find apps quickly on a smartphone with the overwhelmingly large app base, a more efficient and intelligent app launcher is necessary. Nihao, as a time-aware, location-aware, app correlation-aware and personalized app launcher, demonstrates it effectiveness in term of app lookup time and management time. Data analysis result shows that looking up downloaded apps using Nihao yields less time than using the default launcher on average. Then through the user survey, we found that majority of the users thought Nihao could help save the app management time by ranking the apps automatically.

Then we validated that users' app usage pattern are indeed context dependent by showing that Nihao predicts better than the frequency-based predictor. Thus, exploring the correlation between app usage and the richer context is one of our future work.

Finally, we found that the larger UI change does not necessarily yield longer app lookup time as the app lookup time highly depends on the app icon position on screen, which suggests the prediction accuracy is the most important factor in designing such a system. Thus, in the future, we plan to focus on improving the prediction accuracy by experimenting with different models leveraging richer context and testing the system with more users of different backgrounds.

Acknowledgment. This material is based upon work supported partly by the National Science Foundation under Grant No. 1040725 and No. 0917112. Any opinions, ndings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reect the views of the National Science Foundation.

References

1. androlib, http://www.androlib.com
2. Yelp, http://www.yelp.com
3. Hartmann, M.: Challenges in developing user-adaptive intelligent user interfaces. In: Proc. LWA 2009 (2009)
4. Falaki, H., Mahajan, R., Kandula, S., Lymberopoulos, D., Govindan, R., Estrin, D.: Diversity in smartphone usage. In: Proc. ACM MobiSys 2010, San Francisco, CA (June 2010)
5. Bohmer, M., et al.: Falling asleep with angry birds, facebook and kindle - a large scale study on mobile application usage, Stockholm, Sweden (August 2011)
6. Do, T.M.T., Blom, J., Gatica-Perez, D.: Smartphone usage in the wild: a large-scale analysis of applications and context, Alicante, Spain (November 2011)
7. Huang, K., Ma, X., Zhang, C., Chen, G.: Predicting mobile application usage using contextual information. In: Proc. ACM Sagaware 2012 (2012)
8. Xu, Q., et al.: Identifying diverse usage behaviors of smartphone apps, Berlin, Germany (November 2011)
9. Sears, A., Shneiderman, B.: Split menus: effectively using selection frequency to organize menus. ACM Trans. Comput. Hum. Interact. (1994)
10. Findlater, L., McGrenere, J.: Impact of screen size on performance, awareness, and user satisfaction with adaptive graphical user interfaces. In: Proc. SIGCHI 2008 (2008)
11. Phithakkitnukoon, S., Dantu, R., Claxton, R., Eagle, N.: Behavior-based adaptive call predictor. ACM Transactions on Autonomous and Adaptive Systems 6(3) (September 2011)
12. Yan, B., Chen, G.: Appjoy: Personalized mobile application discovery. In: Proc. ACM MobiSys 2011 (2011)
13. Bohmer, M., et al.: Exploring the design space of context-aware recommender systems that suggest mobile applications. In: Proc. CARS 2010 (2010)
14. Yan, T., et al.: Fast app launching for mobile devices using predictive user context. In: Proc. ACM MobiSys 2012 (2012)
15. Google's data interchange format, http://code.google.com/p/protobuf/
16. Play framework, http://www.playframework.org/

Context-Aware Mobile Power Management Using Fuzzy Inference as a Service

Mohammad Moghimi, Jagannathan Venkatesh, Piero Zappi, and Tajana Rosing

University of California, San Diego, Computer Science and Engineering,
9500 Gilman Drive, La Jolla, CA, 92037. USA
{mmoghimi,jvenkatesh,pzappi,tajana}@ucsd.edu

Abstract. As smartphones become ubiquitous, their energy consumption remains one of the most important issues. Mobile devices operate in a dynamically changing context, and their embedded sensors can be used to extract the relevant context needed for resource optimization. In this paper, we present a context-aware power management system implemented as a widely-applicable middleware application. Fuzzy inference is used to represent a high-level description of context, which is provided as a service. We test our approach using actual periodic and streaming applications on a mobile phone. Our results show energy reduction of 13-50% for periodic applications, and 18-36% for streaming applications.

Keywords: Context Awareness, Energy Efficiency, Mobile Systems, Fuzzy Inference.

1 Introduction

Mobile devices such as smartphones have become ubiquitous. As a consequence, they operate in a dynamically changing context, both in terms of user needs and computational requirements. Context has been defined as "any information that can be used to characterize the situation of an entity", and in mobile devices, it is obtained from sensors, communication media, or locally available data [15]. Context awareness in mobile devices has been proven useful when developing user interfaces [15][12], proactive service provisioning [4][5], or efficient resource management [10]. The concept has been explored in several comprehensive studies [2][8][21] over the past decade, as new mobile technologies emerge. In general, previous work on context aware systems focuses either on novel technique for context recognition, or on efficient management of context knowledge. However, one area that has been largely overlooked is leveraging context recognition methods to provide adaptive power management as a service for mobile applications.

The rationale for power management in mobile devices stems from their growth in computational complexity and power consumption with each process generation. However, as mobile battery technology has not grown at the same rate, device lifetimes are decreasing [19]. Subsequent work [3] analyzes power consumption in Android-based smartphones during common tasks, noting the significant contribution

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 314–327, 2013.

of GPRS and Wi-Fi radios, graphics processors, and the CPU. Consequently, substantial research has been devoted to leveraging context to reduce power consumption or manage battery use. In [1] the authors aim to optimize the high energy overhead in mobile networking by leveraging characteristics of typical mobile applications, scheduling transfers in batches to reduce the overhead of each transfer, and aggressively prefetching data for applications that can benefit from it. Similarly, the work in [20] curbs the energy consumption of data transmission over Wi-Fi or cellular networks by exploiting the complementary energy traces of each type of connection. The application opts to use either Wi-Fi or cellular data transmission by determining the lower-energy operation based on the current context. The authors demonstrate an improvement in battery lifetime by up to 42%. The work presented in [19] uses energy traces from applications and location information to determine when and where phones can be charged, and how much remaining energy is available for critical applications. Consequently, the energy manager allocates resources for non-critical applications based on the time to the next predicted charging opportunity. Musolesi et al [17] investigates the high energy consumption of continuous location sensing applications which need to communicate GPS data with a backend server. Using short-term prediction of movement based on context, the application duty-cycles communication with the backend, demonstrating up to 60% communication savings with limited impact on accuracy. Similarly, [13] and [24] leverage context to determine how often to retrieve GPS location, opting instead to use the adaptive frequency or other means of determining location for energy savings.

The related work above opts to exploit context awareness either in a limited fashion, or for a specific application. However, mobile platforms such as Android, Windows Phone, and iOS run many applications at a time. Competition among running applications for access to sensor data in order to independently retrieve similar context information is both inefficient and redundant. A better approach is an intermediate, extensible layer which can retrieve input from the many sensors and sources of raw context available on phones. In turn, it would consolidate the input into higher-level, filtered, and processed context that is queryable and usable by all applications.

The advantage of this approach is the elimination of redundancy in each application, which would otherwise need to independently process context data, as well as removing the constraint of sharing sensor access among applications. Previous work approached the concept of context as a service for goals other than power management. For example, the context manager designed in [15] retrieves raw context information from sensors, offering high-level context aimed at providing an adaptive UI. Similarly, [11] designs and implements a context delivery system for Symbian devices. The end-to-end implementation provides context as a middleware service, abstracting raw sensor input into usable context via fuzzy inference. The resulting context is made available to applications as a query, via subscription to specific updates, or through even higher abstracted services, which perform post-processing using simple Bayesian networks for more complex context. The system exemplifies the type of context management we aim to leverage in our work. We aim to extend this approach further by providing context-aware actions to applications in a power-aware, adaptive user experience.

Typical context aware systems use decision tree-based methods in their inference block [10] [20]. Decision trees are a generalization of IF-THEN-ELSE decisions, whose outputs represent the actions to be taken. Rules can be as simple as a timeout from the last screen interaction to more complex rules combining the interaction of data from multiple sensors. This approach establishes crisp thresholds on context variables, which can cause extreme or unintentional responses to complex input. In contrast, fuzzy inference handles context in a more natural way. Since mobile system context is derived in part from environmental behavior, it is inherently fuzzy: described using degrees of confidence rather than absolutes [15]. Additionally, as context recognition involves a degree of uncertainty due to sensor inaccuracies or the variability in defining a context, the use of exact thresholds is not always reasonable. As such, fuzzy logic variables have a truth value or a degree of confidence ranging from 0 and 1, which is a more natural expression of typical context information. Fuzzy inference provides an additional advantage: the aggregation of different rules based on confidence values expressed in a continuous manner, which is difficult when using a decision tree or discrete implementation. As higher abstractions of context can determined by the correlation of variables with different, overlapping confidence factors, fuzzy logic provides a natural means to aggregate raw input data into appropriate context. Due to its advantages, fuzzy logic has been used extensively for context detection in mobile devices as well as other embedded systems. Mäntyjärvi et al. [11][15] design and implement a context manager based on fuzzy inference to provide adaptive UI management for Symbian devices. Ciaramella et al. [4] develop an application selection service using fuzzy logic that incorporates location, time, and perceived task (based on applications running and preset situations). The contexts are provided as a query to a backend service which returns the most recommended application to achieve the perceived task. Martinez et al. [16] propose a fuzzy logic system for energy management in hybrid vehicles, referencing the ability of fuzzy systems to outperform their counterparts in embedded domains, as well as their usefulness for energy management. Context variables represent battery state-of-charge, supercapacitor state, terrain, and driving conditions, and use human experience to determine which power subsystem should propel the vehicle at a given time.

In this paper, we leverage such context managers to provide adaptive power management for mobile phones. We design and implement a low complexity, extensible framework that uses fuzzy inference of power-related context variables to mitigate a significant problem in mobile devices: energy consumption. Applications expose needed context parameters to the power manager and register rules for adaptive behavior. In turn, the power manager tunes applications parameters based on the rules and the available context. We test our power manager on two applications that are representative of typical mobile tasks. The first represents applications that run in the background, periodically retrieving data from a remote sever (e.g. RSS feed, email). We define a cost function that models the trade-off between energy consumption of the system and its performance (measured as delay in presenting the user desired information) with which we can compare our approach to decision tree systems. We also develop and test a representative of streaming applications

(e.g. audio and video streaming), which require continuous, lossy data retrieval from a backend server. We define a threshold for audible bitrates, and measure power savings on the device under context-aware conditions in comparison to non-adaptive conditions.

The rest of the paper is organized as follows. Section 2 describes the proposed power management method. Section 3 outlines the applications developed, the experiments performed, and their results. Section 4 concludes with possible future directions.

2 System Design

In this section, we present an overview of the system, including the implementation of fuzzy logic and exploring the major components of the power management system.

2.1 Fuzzy Inference

Fuzzy logic is a generalization of set theory, which, instead of binary membership, uses functions that assign every input a value between 0 and 1. The outputs, or context variables, can be used to provide a high-level abstraction of low level data. For example, time contexts such as *TimeOfDay* = {*Morning, Work Hours, Evening, Night*} represent a high-level abstraction of the low-level input of system time. Figure 1 depicts the member functions that are composed to form the fuzzy logic context for *TimeOfDay*.

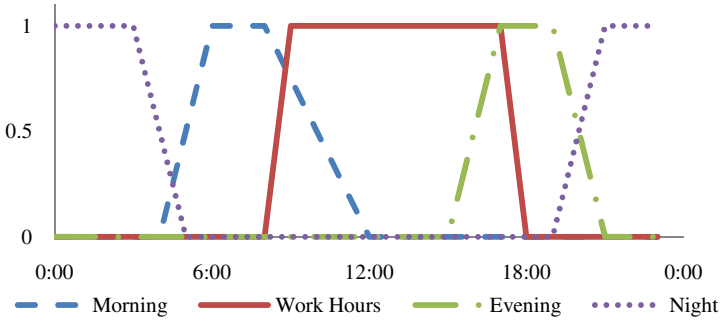


Fig. 1. Fuzzy membership functions for *TimeOfDay*. In our experiments, the values for *Morning*, *Work Hours*, *Evening*, and *Night*, are determined statically, though this can be determined through machine learning models and other methods as well.

Client applications can take advantage of the more gradual transitions in fuzzy logic to develop more gradual control of system behavior. In the above example,

Night is determined by the period in which night is detected with absolute confidence ($x \leq 3 \cup x > 21$) as well as a gradual threshold for when night begins ($19 < x \leq 21$) and ends ($3 < x \leq 5$).

$$\mathbf{Night} = \begin{cases} \mathbf{1}, & \boldsymbol{x} \leq \mathbf{3} \cup \boldsymbol{x} > 21 \\ \frac{5}{2} - \frac{1}{2}\boldsymbol{x}, & \mathbf{3} < x \leq 5 \\ \frac{1}{2}\boldsymbol{x} - \frac{19}{2}, & \mathbf{19} < x \leq 21 \\ \mathbf{0}, & \mathbf{5} < x \leq 19 \end{cases} \tag{1}$$

This example illustrates a fundamental advantage of fuzzy logic for use with context: there is not necessarily a pure definition of *Night*, so instead of attempting to force such an absolute definition, we instead define the boundary around night with a growing confidence.

Defining the optimal thresholds for each member of *TimeOfDay* is highly dependent on the behavior of the user. While in this case, these boundaries are defined statically, the fuzzy context management systems explored in Section 1 defined thresholds statically or through learned behavior.

2.2 Context-Aware Power Manager

The power management system we develop is designed to be flexible and extensible, providing context-aware, adaptive behavior, to applications. This approach strikes a balance between individual applications retrieving sensor information independently [2], and a central context management system that registers updates with every application that requests it [11]. In both cases, the burden of continually adapting to context changes is placed on the applications themselves. Additionally, in the former case, the Context Detection Block (CDB) can become a bottleneck for applications attempting to retrieve context, and inefficient if they try to retrieve the same context information. For example, two applications that need the same context variable (i.e. the time of day, or the ambient noise level) must separately be updated every time the context changes. Instead, we place a power manager in between the application and the CDB, which serves a dual purpose. First, it aggregates power-relevant context variables from both the sensors directly, and from a CDB, which provides higher abstractions of raw context. This removes the inefficiency of applications retrieving the same context. Secondly, instead of performing actions independently, applications register rules for updating context-dependent input data with the power manager.

This allows similar applications to adapt to context with the same variables. The resulting system, composed of sensor input, CDB, and the power manager, forms a flexible framework, able to use any context detection block, including those specified in Section 1. Figure 2 shows the architecture of the framework.

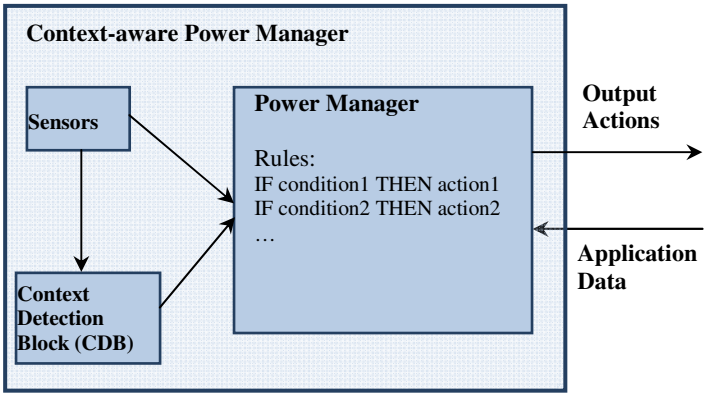


Fig. 2. Layout of the *Context-aware Power Manager*, showing *sensors* for raw data, a *context-detection block* for processed data, and the *power manager* with sample rules

The power manager, unique to our system, retrieves context information and outputs a set of application-specific actions. Context information is either derived directly from input sensors (for simple contexts such as time or date), or as high-level abstractions in the context detection block (for processed context such as relative location, noise level, or user activity level). The output actions are application-specific variables, registered by the respective applications. The power manager tunes these variables based on static rules to provide context-aware energy-efficient behavior.

For example, let us consider a periodic application whose rate of download (*UpdateRate*) can be modified by the power manager based on the time of day. The rules governing the update can simply use numerical (via system time), or a more fuzzy representation, such as the *TimeOfDay* variable discussed above (provided by the CDB). Such a rule can be paired with the output action, i.e. *UpdateRate*={10min, 30min, 60min, 120min}. A rule connecting *TimeOfDay* with *UpdateRate*, can be specified in the power manager in the format "IF context THEN output_action". In the case of *TimeOfDay*, we can establish the update interval of a periodic application based on the intuition that users do not check an application for updates at night:

IF *TimeOfDay* == "night", **THEN** *UpdateRate***=** "1hr-update"

This further illustrates the advantage of registering rules in the power manager: different periodic applications that desire to use the same action can all register to use UpdateRate to determine their output action, reducing their query rate to once per hour.

The system, implemented for Android API Version 10, is designed to be versatile and extensible. The power manager can retrieve new raw (from sensors) or abstracted (from the context detection block) context variables, extend and re-aggregate existing context variables, and establish new or extend rules for output actions.

Our system accomplishes all such goals with limited interaction with the established platform. Each application provides two descriptor files to the power manager: a fuzzy control logic (FCL) file and a properties file. The FCL file [9]

exposes context-sensitive application parameters that can be tuned, and establishes the rules for adapting to the context. More specifically, the FCL file lists the context variables needed, the output actions modified, the fuzzy sets that compose both, and the piecewise linear membership functions which translate input context to the output actions. The properties file is used to store updates to the context-dependent variables for an application. Figure 3 shows the overview of the system and the interaction between the components.

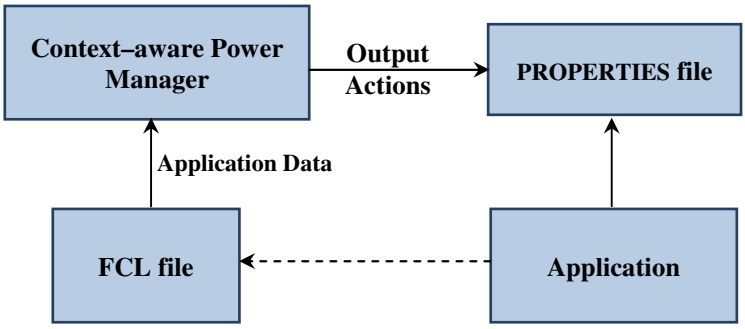


Fig. 3. Software component overview, consisting of the *Context-aware Power Manager* (see Fig. 2), the *FCL file* that provide application data, and the *PROPERTIES file*, to which output actions are published, and which is read by the *application*.

When a context-aware application is developed, it registers with the power manager by exposing its FCL and properties files. The power manager parses the FCL file, registering any new context inputs and action outputs, and integrates the rules needed to translate the context into the member functions. When the power manager is running, it provides the defuzzified output variables, which determine the application's adaptations, to the properties file. For example, in the case of streaming media applications, context would be used to determine the bitrate of the incoming stream, which strongly affects the energy consumed by the device. The power manager would output the new bitrate for the application to the properties file, which then changes the quality of the incoming stream as desired.

3 Experiments and Results

3.1 Experimental Setup

We implemented our framework on both a T-Mobile G1 development phone and a Nexus One phone. Power measurements for the running applications are provided via the PowerTutor application. PowerTutor estimates power consumption based on a power model customized for each phone's hardware, and has been demonstrated to have less than 2.5% long-term error [23]. Our experiments consisted of two separate classes of applications: *periodic downloading applications*, which query and retrieve data from a backend at specific intervals, and *streaming applications*, which need to constantly retrieve data over the duration of the stream.

3.2 Periodically Downloading Applications

Periodically downloading applications have risen in popularity, and now consist of a significant portion of mobile apps [5]. We model our test applications after common examples of such apps: RSS feed-following, Twitter, and e-mail retrieval, implemented in RSSApp, TwitterApp, and EmailApp, respectively. Each application is implemented as a background service without a graphical user interface (GUI). Communication to the backend composes a majority of the energy requirement for such applications, and consequently, our goal with each of the model applications is to reduce the amount of energy spent on connectivity. The work in [1] analyzed the energy cost of such network-dependent applications, identifying tail energy, caused by mobile data connections' retention of high-energy states even after completion of data retrieval, to be a significant contributor to energy use. To compensate, the authors implemented batch-processing of periodic applications to avoid recurring overheads, and a similar approach was undertaken in [14]. We approached the problem in a similar manner, with the added advantage of the context framework.

Observed user interaction with periodic applications shows a variation over the course of the day, requiring more frequent updates in the morning, as the user is brought up to date on the events of the recent past, and less frequent updates throughout the day, culminating in low access needs at night. Additionally, we leverage the lower energy requirements of Wi-Fi over GSM/GPRS connectivity.

We establish *DayType* = {Weekday, Weekend}, *TimeOfDay* = {Morning, Noon, Evening, Night}, and *WifiAccess* = {Wi-Fi, No Wi-Fi} as context variables, and *TimeStep* = {Short, Medium, Long} and *levelOfInfo* = {0, 1, 2} as the output actions. *TimeStep* refers to the update frequency of the application, *levelOfInfo* refers to the depth of data needed by the application (0 = title only, 1 = title and text, and 2 = title, text, and attachments), and *WifiAccess* = represents the access to Wi-Fi connectivity. The fuzzy context rules are described in Table 1 and Table 2.

Table 1. TimeStep membership sets for *DayType* and *TimeOfDay*

	Morning	**Noon**	**Evening**	**Night**
Weekday	Short	Short	Medium	Long
Weekend	Medium	Medium	Medium	Long

Table 2. LevelOfInfo membership sets for *WifiAccess* and *TimeOfDay*

	Morning	**Noon**	**Evening**	**Night**
Wi-Fi	2	1	1	0
No Wi-Fi	2	2	2	0

The defuzzification of *TimeStep* is obtained by the center of gravity (cogs) for the piecewise functions that comprise Short, Medium, and Long, each weighted by the level of confidence in the particular function. This is defined in Equation 2:

$$\mathbf{f} = \mathbf{cogs}(\mathbf{C_{short}} * \mathbf{f_{short}} \cup \mathbf{C_{med}} * \mathbf{f_{med}} \cup \mathbf{C_{long}} * \mathbf{f_{long}}) \,. \quad (2)$$

where Short, Medium, and Long denote the piecewise functions that compose each set, and C_{short}, C_{med}, and C_{long} representing the confidence for each set, respectively. The output, f is the numerical output representing the period for updating the application.

The experiments for the RSSApp and TwitterApp use feed traces from mashable.com, a technology news feed, and the EmailApp used a Gmail account to retrieve email messages. User interaction is simulated on three user models: Heavy User, Medium User, and Light User, which are derived from models presented by Falaki et al. in [6]. The delay time is measured from the time of the request to the time of the response.

Table 3 summarizes the actual power consumption and the delay for different adaptation techniques: fuzzy inference, a simple decision-tree model, and a fixed update rate representing no adaptation. The decision tree model uses crisp thresholds for each state, defining the action to take as a list of conditionals.

Table 3. Comparison of adaptive techniques for periodic downloading applications

	Fuzzy Inference		**Decision Tree**		**Fixed**	
	Avg. Power (W)	***Delay (s)***	***Avg. Power (W)***	***Delay (s)***	***Avg. Power (W)***	***Delay (s)***
RSS App	0.345	9.94±3.0	0.390	10.80±2.6	0.658	6.34±2.3
Twitter App	0.289	7.33±2.4	0.330	9.40±1.9	0.780	5.9±1.0
Email App	0.510	8.11±0.4	0.500	14.00±2.4	0.800	10±1.1

The energy consumption in all three periodic applications is significantly reduced due to context-aware adjustment of both the frequency of the data retrieved and the level of data downloaded. However, longer *TimeStep* values and lower *levelOfInfo* increase the chance that users experience a delay in retrieving backend data or the level of data they require. Fuzzy logic demonstrates, on average, a 49% improvement in average application power reduction over no adaptation, although with 14% increase in retrieval delay. Additionally, fuzzy logic demonstrates, on average, 12% improvement in power reduction for the RSSApp and the TwitterApp, and a reduction in delay of 14.5%. For the EmailApp, the decision tree method performed slightly better in terms of power consumption (2%), though with 42% increase in delay. An analysis of the results shows that the decision tree's crisp boundaries allow it to spend more time in the longest *TimeStep*, whereas the smoother transitions of fuzzy inference result in more frequent intermediate *TimeSteps*. However, the slight power saving comes at a high delay cost. While the delay cost is a qualitative measure, fuzzy inference still provides an improvement over the decision tree model, with a significant improvement in delay.

We then evaluate the tradeoff between the time-delay needed to retrieve the necessary data and the energy cost for retrieving data. We test the three user models: light, medium, and heavy, where each model indicates the frequency of application usage based on time and day. In order to represent the usage pattern of a mobile user, we can expect a fixed average rate of use, with the average number varying between light, medium, and heavy. As each device usage instance is independent, one

representation of usage patterns is a Poisson distribution, which has previously been published in [7]. Figure 3 depicts the tradeoff between power usage and the average delay that a user faces to get updated data for the different usage patterns. The average delay stabilizes near 5 seconds as usage increases, regardless of the increase in *UpdateTime*. The result helps illustrate two ways to optimize the functionality of the power manager: utilizing learning algorithms to limit thresholds, and statically analyzing an application so that appropriate limits may be set to minimize power consumption.

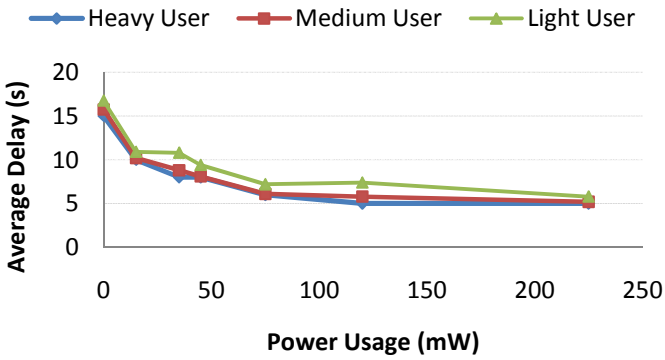


Fig. 4. The delay vs. power usage for periodically downloading applications

3.3 Streaming Applications

Streaming applications are common in the mobile world, with the prevalence of services like Pandora and Google Music. We model our test application after a streaming music service, allowing streaming playback of a remote playlist. Similar to periodic applications, data retrieval with the backend service composes a majority of the energy consumption for streaming applications, although real-time playback precludes the use of batch information retrieval. Instead, we leveraged available context to vary the bitrate of data recovery, reducing the amount of data consumed by the running application over its lifetime.

In designing the context awareness of the application, we take advantage of the notion that a quiet environment enables reducing the quality of the streaming music, as the user is less likely to increase the volume and become exposed to distortion. Conversely, in when the environment is noisy, a higher quality stream mitigates distortion at higher volumes. Similarly, we perform a simple activity analysis from accelerometer readings. If the user is stationary, the quality of the stream is likely more important, as the music is a primary activity. If the user is walking or running, however, he or she is most likely listening to music as a secondary activity, and listening to music is a secondary activity.

As such, we define *AmbientNoise* = {*Quiet*, *Moderate*, *Loud*} and *Activity* = {*Stationary*, *Walking*, *Running*} as context variables, and *Bitrate* = {*Very Low*, *Low*, *Medium*, *High*, and *Very High*} as the output actions. The fuzzy context rules are described in the table below.

Table 4. Bitrate member sets for *AmbientNoise* and *Activity*

	Stationary	**Walking**	**Running**
Quiet	Medium	Low	Very Low
Moderate	High	Medium	Low
Loud	Very High	High	Medium

The experiments used accelerometer and microphone traces from a 1-hour period of activity which involved all three *Activity* types and all three *AmbientNoise* levels. Accelerometer amplitudes and directions were converted to abstracted activity contexts using a variation of the algorithms presented in [22], simplified to reflect only walking, running, and remaining stationary. Microphone input was translated to quiet, moderate, and loud using a training set to capture minimum and maximum volume levels, and dividing the range into three equal sets. A separate program is used to stitch together a variable-bitrate file based on the traces and Equation 3 below. The resulting file is then streamed to the phone for the experiment. The defuzzification of the output settings is defined by the linear combination of confidence of settings associated with each member:

$$\begin{aligned} f = f_{med} * \frac{(C_{stat} + C_{quiet}) + (C_{walk} + C_{mod}) + (C_{run} + C_{loud})}{6} \\ + \mathbf{f_{high}} * \frac{(\mathbf{C_{stat}} + \mathbf{C_{mod}}) + (\mathbf{C_{walk}} + \mathbf{C_{loud}})}{\mathbf{4}} \\ + \mathbf{f_{low}} * \frac{(\mathbf{C_{walk}} + \mathbf{C_{quiet}}) + (\mathbf{C_{run}} + \mathbf{C_{mod}})}{\mathbf{4}} \\ + \mathbf{f_{v.low}} * \frac{(\mathbf{C_{run}} + \mathbf{C_{quiet}})}{\mathbf{2}} + \mathbf{f_{v.high}} * \frac{(\mathbf{C_{stat}} + \mathbf{C_{loud}})}{\mathbf{2}} . \end{aligned} \quad (3)$$

where the $f_{v.low} = 128$kbps, $f_{low} = 256$kbps, $f_{med} = 384$kbps, $f_{high} = 512$kbps, and $f_{v.high} = 640$kbps. The selected frequencies for each state were determined by linear growth from the lowest to the highest available frequencies {128,192,…,640}, and the resulting frequency is discretized to the closest member in the frequency range.

Table 5. Comparison of adaptive techniques for streaming applications

	Fuzzy Inference		**Decision Tree**		**Fixed**	
	Avg. Power (W)	***Median bitrate (kbps)***	***Avg. Power (W)***	***Median bitrate (kbps)***	***Avg. Power (W)***	***Median bitrate (kbps)***
Streaming App	0.689	256	0.844	256	0.983	512

We again perform a comparison of fuzzy inference with both a decision tree model and a fixed (non-adaptive) model. The results in Table 5 show marked improvement, with fuzzy inference providing, on average, 29.9% improvement over a non-adaptive application and 18.4% improvement over a decision tree model. The improvement over the decision tree model is most striking, and is attributed to the more natural, intermediate transitions enabled by confidence values between states. The discrete sets of the decision tree force sudden, large transitions which do not take advantage of potential intermediate transitions between, for example, Stationary+Moderate and Walking+Quiet. As such, the decision tree spends a longer time in the higher-bitrate state and consumes on average more power. Figure 5 displays the results of one of the experiments, and helps illustrate the gradual vs. stark transitions between the fuzzy inference and decision tree methods for a subset of the streaming experiment. For example, the raw sensor data shows a steady increase in ambient noise between minutes 4 and 10. Both the decision tree and the fuzzy inference models react to this change by increasing the bitrate accordingly. However, the sharp, early increase in the decision tree model contrasts with the more gradual change of fuzzy logic for the same period. Additionally, at time t=5, the crisp threshold of the decision tree model jumps to 512kb, implying the need for a High bitrate. However, fuzzy inference demonstrates that there is actually not a need for a High bitrate due to there not being enough confidence for that state, choosing instead to gradually increase with increasing ambient noise until leveling off at 448kbps. Similarly, the decision tree model increases to 640kbps at t=11, though fuzzy inference demonstrates that a lowered confidence in the Very High state should actually limit the value to 576kbps. Both these cases exemplify the advantage of fuzzy logic in adapting to context and, more importantly, its ability to better reflect the more graduated changes that actually occur in the surrounding context.

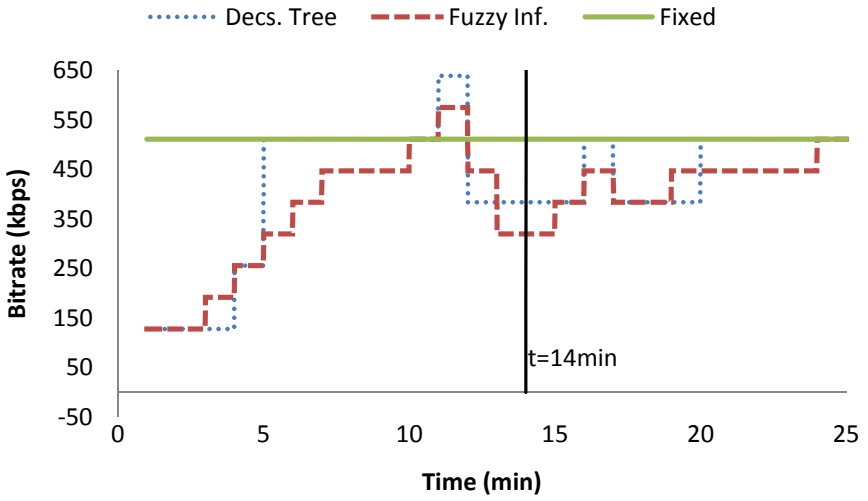


Fig. 5. Experimental bitrate results for Streaming Applications. At t=14, the decision tree model was streaming at 384kbps, the fuzzy logic model was streaming at 320kbps, compared with a fixed stream at 512kbps. Time t=4 to t=10 demonstrates the difference in response for decision tree compared to fuzzy logic in adapting to gradually increasing ambient noise levels.

4 Conclusion

In this paper, we introduce the idea of using context information to minimize energy usage of characteristic applications on mobile devices. We explore the background research in context awareness, and illustrate the advantages fuzzy inference of context provides to both mobile applications and in energy efficiency. We then propose and implement a context-aware power manager on mobile phones leveraging both raw and abstract context detection, and fuzzy inference to adapt typical mobile applications' behavior to current context. The proposed technique has been compared to similar method and showed 10 to 50% percent lower power consumption, as well as up to 18% improvement over traditional decision-tree models. Future work includes expanding the power manager with additional context variables and applications, as well as using learning techniques to dynamically determine higher-level context, as opposed to fixing them with static values. This has been utilized in related work for determining context [13][18], and allows abstract context variables to be adapted to the user.

Acknowledgements. This research was supported by NSF Grant CNS-0932403, the Center for Networked Systems (http://cns.ucsd.edu), and Qualcomm.

References

1. Balasubramanian, N., Balasubramanian, A., Venkataramani, A.: Energy Consumption in Mobile Phones: A measurement Study and Implications for Network Applications. In: Proc. of 9th Conf. on Internet Measurement, pp. 280–293 (2009)
2. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. Pervasive and Mobile Computing 6(2), 161–180 (2010)
3. Carroll, A., Heiser, G.: An analysis of power consumption in a smartphone. In: Proc. USENIX, p. 21 (2010)
4. Ciaramella, A., Cimino, M., Lazzerini, B., Marcelloni, F.: Situation-Aware Mobile Service Recommendation with Fuzzy Logic and Semantic Web. In: Proc. 9th Int. Conf. on Intelligent Systems Design and Applications, pp. 1037–1042 (2009)
5. Chen, G., Kotz, D.: Solar: An Open Platform for Context-Aware Mobile Application. In: 1st Int. Conf. on Pervasive Computing, pp. 41–47 (2002)
6. Falaki, H., Mahajan, R., Kandula, S., Lymberopoulos, D., Govindan, R., Estrin, D.: Diversity in smartphone usage. In: Proc. 8th Int. Conf. on Mobile Systems, Applications and Services, pp. 179–194 (2010)
7. Gündüz, Ş., Özsu, M.T.: A Poisson Model for User Accesses to Web Pages. In: Yazıcı, A., Şener, C. (eds.) ISCIS 2003. LNCS, vol. 2869, pp. 332–339. Springer, Heidelberg (2003)
8. Hong, J., Suh, E., Kim, S.: Context-aware systems: A literature review and classification. Expert Systems with Applications 36(4), 8509–8522 (2009)
9. International Electrotechnical Commission. IEC 1131- Programmable Controllers Part 7 - Fuzzy Control Programming (1997)
10. Kim, K., Min, A., Gupta, D., Mohapatra, P., Singh, J.: Improving Energy Efficiency of Wi-Fi Sensing on Smartphones. In: Proc. of IEEE Int. Conf. on Computer Communications (2011)

11. Korpipaa, P., Mantyjarvi, J., Kela, J., Keranen, H., Malm, E.: Managing Context Information in Mobile Devices. IEEE Pervasive Computing 2(3), 42–51 (2003)
12. Lemlouma, T., Layaida, N.: Context-aware adaptation for mobile devices. In: Proc. of IEEE Int. Conf. Mobile on Data Management, pp. 106–111 (2004)
13. Lin, K., Kansal, A., Lymberopoulos, D., Zhao, F.: Energy-Accuracy Trade-off for Continuous Mobile Device Location. In: Proc. 8th Int. Conf. on Mobile Systems Applications and Services, pp. 285–297 (2010)
14. Mahesh, M., Calder, M.: Batch Scheduling of Recurrent Applications for Energy Savings on Mobile Phones. Sensor Mesh and Ad Hoc Communications and Networks, 1–3 (2010)
15. Mäntyjärvi, J., Seppänen, T.: Adapting applications in handheld devices using fuzzy context information. Interacting with Computers 15(4), 521–538 (2003)
16. Martínez, J., John, R., Hissel, D., Péra, M.: A survey-based type-2 fuzzy logic system for energy management in hybrid electrical vehicles. Information Sciences 190(1), 192–207 (2012)
17. Musolesi, M., Piraccini, M., Fodor, K., Corradi, A., Campbell, A.T.: Supporting Energy-Efficient Uploading Strategies for Continuous Sensing Applications on Mobile Phones. In: Floréen, P., Krüger, A., Spasojevic, M. (eds.) Pervasive 2010. LNCS, vol. 6030, pp. 355–372. Springer, Heidelberg (2010)
18. Paek, J., Kim, J., Govindan, R.: Energy-Efficient Rate-Adaptive GPS-based Positioning for Smartphones. In: Proc. 8th Int. Conf. on Mobile Systems Applications and Services, pp. 299–314 (2010)
19. Ravi, N., Scott, J., Lu, H., Iftode, L.: Context-aware Battery Management for Mobile Phones. Pervasive Computing and Communications, 224–233 (2008)
20. Rahmati, A., Zhong, L.: Context-for-Wireless: Context-Sensitive Energy-Efficient Wireless Data Transfer. In: Proc. 8th Int. Conf. on Mobile Systems Applications and Services, pp. 165–178 (2007)
21. Shye, A., Scholbrock, B., Memik, G.: Into the wild: studying real user activity patterns to guide power optimizations for mobile architectures. In: Proc. 42nd Int. Sym. on Microarchitecture, pp. 168–178 (2009)
22. Weiss, G., Kwapisz, J., Moore, S.: Activity Recognition using Cell Phone Accelerometers. ACM SIGKDD Explorations, 74–82 (2010)
23. Zhang, L., Tiwana, B., Qian, Z., Wang, Z., Dick, R., Mao, Z., Yang, L.: Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In: Proc. 8th Int. Conf. on Hardware/Software Codesign and System Synthesis, pp. 105–114 (2010)
24. Zhuang, Z., Kim, K., Singh, J.: Improving Energy Efficiency of Location Sensing on Smartphones. In: Proc. 8th Int. Conf. on Mobile Systems Applications and Services, pp. 315–330 (2010)

Automatic Annotation of Daily Activity from Smartphone-Based Multisensory Streams

Jihun Hamm[1], Benjamin Stone[2], Mikhail Belkin[1], and Simon Dennis[2]

[1] The Ohio State University, Dept. Computer Science and Engineering, Columbus, OH 43210, USA
[2] The Ohio State University, Dept. Psychology, Columbus, OH 43210, USA

Abstract. We present a system for automatic annotation of daily experience from multisensory streams on smartphones. Using smartphones as platform facilitates collection of naturalistic daily activity, which is difficult to collect with multiple on-body sensors or array of sensors affixed to indoor locations. However, recognizing daily activities in unconstrained settings is more challenging than in controlled environments: 1) multiples heterogeneous sensors equipped in smartphones are noisier, asynchronous, vary in sampling rates and can have missing data; 2) unconstrained daily activities are continuous, can occur concurrently, and have fuzzy onset and offset boundaries; 3) ground-truth labels obtained from the user's self-report can be erroneous and accurate only in a coarse time scale. To handle these problems, we present in this paper a flexible framework for incorporating heterogeneous sensory modalities combined with state-of-the-art classifiers for sequence labeling. We evaluate the system with real-life data containing 11721 minutes of multisensory recordings, and demonstrate the accuracy and efficiency of the proposed system for practical lifelogging applications.

Keywords: mobile computing, lifelogging, activity recognition, automatic annotation.

1 Introduction

With an ever increasing number of smartphones today which are capable of recording motion, location, vision and audio, there are numerous applications that can make use of the multisensory data, such as context-aware services, health monitoring, augmented memory, and lifelogging. Lifelogging refers to a long-term process of automatically collecting sensory data using wearable devices, and storing the data into a personal multimedia form for browsing, annotating, and searching [14,20,39,7,10]. A lifelogging system will be particularly useful if it is capable of recognizing high-level experiences of users from low-level sensory streams without requiring the user to manually annotate the huge amount of data. In this paper, we present a lifelogging system for capturing a user's daily experience by collecting multisensory streams on a smartphone and automatically annotating the daily activity with high-level tags.

Automatic annotation is closely related to activity recognition, which has a rich literature across various fields including embedded and mobile systems, pervasive and ubiquitous computing, sensor networks, multimedia, intelligent systems, and machine

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 328–342, 2013.

learning. (We refer the reader to [22,23] for a survey.)[1] The majority of previous research on activity recognition used custom on-body sensors and embedded systems [3,25,32,48,8,17,1,26,10,24,36], or used external sensors such as surveillance cameras/microphones, object-attached sensors, and RFID tags [27,42,43,34,16,19].

Recently, some researchers have focused on using smartphones as a platform [4,7,12,44,49], which allows them to collect naturalistic data from a user who carries out everyday activity without interruption. The range of activities resulting from such a setting is much broader that atomic sets of actions performed by a user in a controlled setting such as postures/locomotion-types (sitting, lying, walking, running, standing up, sitting down, up/down the stairs, etc), and opens up new opportunities for studying everyday human behavior.

However, recognizing daily activities in unconstrained settings is more challenging than in controlled environments in two aspects – sensory data and activity labels. Firstly, various sensors equipped in generic smartphones are noisier and have to operate under a limited battery capacity, resulting in sparser samples than those from a surveillance system for example. More importantly, multisensory streams from heterogeneous sensors can be asynchronous, vary in sampling rates and have missing data occasionally. In this work, we propose a multisensory bag-of-word representation to handle these problems. The bag-of-words model, originally used for document analysis, has proven useful in other domains such as visual scene analysis [13] and activity recognition in particular [17,7,44]. We build on the previous work and present our multisensory bag-of-word framework that can be combined with various classification algorithms in the subsequent stage.

The second challenge with unconstrained daily activities is that they are continuous, can occur concurrently, and have fuzzy onset and offset boundaries. Furthermore, since the ground-truth labels for activities cannot be acquired on the fly but only after the collection by the user who reviews the logged images at the end of each day, the resultant ground-truth labels are prone to errors and are accurate only in a time scale of minutes rather than seconds or milliseconds. Various classifiers have been used for activity recognition including Fuzzy Logic, Neural Network, Naive Bayes, Bayesian Network, Nearest Neighbor, Decision tree, Support Vector Machines, boosting and bagging (refer to [23]). More recently, continuous recognition of activity was posed as a sequence labeling problem [37,41,46,9,45,31], using temporal models such as Hidden Markov Model [35]), Conditional Random Field (CRF)[21], and structured Large-Margin classifiers[2,40]. The continuous nature of daily activity makes the temporal models potentially more appropriate for handling noisy multisensory streams and labels from smartphones.

In this paper, we describe our system for acquiring naturalistic data from smartphones, and present the multisensory bag-of-words framework combined with state-of-the-art classifiers for automatic annotation of daily activity. We evaluate our approach using 42 days (corresponding to 11721 minutes) of recordings from a volunteer by comparing the performance of various classifiers that represent temporal vs non-temporal and generative vs discriminative approaches, and demonstrate the feasibility of automatic annotation of unconstrained daily activity.

[1] We omit the discussion of activity recognition approaches based on continuous videos.

The rest of the paper is organized as follows. In Section 2, we describe our lifelogging system and its components. In Section 3, we present our framework for handling multisensory streams. In Section 4, we introduce several generative and discriminative approaches for activity recognition. In Section 5, we describe the experiments and report evaluation results, and conclude the paper in Section 6.

2 System

Our system consists of a mobile app, a server infrastructure, and a user interface. We describe them in the following sections.

2.1 Mobile Application and Server

We developed a Java app that acquires multisensory data on Android-based smartphones. The app acquires image, audio, GPS, accelerometer, and other information in regular but changeable time intervals, and stores them until it connects to the server. It runs in the background as a service so as not to disturb normal usage of the phone, although we used the smartphones only for data acquisition purposes in this paper.

Users carried the phone daily from morning till evening. The phone was carried inside a pouch attached to a neck strap to allow an unobstructed view for the camera. The app runs continuously on a standard Android phone for six hours before running out of battery, and it can last much longer with an extended battery. The collected data is sent automatically to a remote server, usually once a day in the evening, when the phone detects WiFi and is connected to a charger. The data is sent in batch mode via SFTP protocol for added security and remains inaccessible to other users in the system.

2.2 Sensory Data and Raw Features

Accelerometer. We use a tri-axial accelerometer with a maximum sampling of 16 Hz. The actual sampling rate obtained from the phone varies over time, so we took only contiguous samples whose actual rates are within a tolerable range. In the literature, simple time-domain features (mean, variance, zero-crossing rate, autocorrelation, etc) and especially frequency-domain features (FFT, spectral entropy, etc) are used for activity recognition [3,25,26,16]. We also perform 16 sample-long FFT on the accelerometer signals from each axis to get a sequence of 27-dimensional raw features.

GPS. The 2D GPS coordinates are obtained after a picture is taken from the camera, and therefore have a similar sampling rate as the images ($\sim$ 1 per minute). The GPS unit is turned off after acquiring the coordinates to preserve battery life and has to lock-on to satellite signals each time. The coordinates are often unavailable due to the failure to lock on inside a building, and they are treated as missing data.

Image. We use 24-bit color JPEG images of size 480 x 640, although much higher resolution is available in current phones. To handle a large amount of data per day, we took a sparse number of pictures ($\sim$ 1 per minute). Original images are stored in the server for lifelogging purpose, but they cannot not be used directly in analysis for privacy reasons.

Table 1. List of 30 tags in three categories describing daily activity

Category	Tags
Activity	other activity, walk, drive/inside a vehicle, eat/drink, talk/chat/discuss, chores (cook/clean/laundry/etc), exercise/play sports, listen to a lecture, give a lecture, shop in a store, tend to baby, use a computer, watch tv/movie, pick up/drop off, read/write on paper/board
Place	other place, my home, my office, classroom/meeting room, other's office, restaurant/cafe, store, public places, outdoor
People	other people, my family, friend(s), colleague(s), stranger(s), crowd

A simple feature extraction can be performed to remove identity-revealing information from the images. Kim et al [20] used CIELAB-space color map and orientation map to find salient regions, and then computed SIFT features [28] as visual descriptors. Doherty et al [11] used scalable color and edge histogram, which is a subset of MPEG-7 feature descriptors [5]. We opt for the latter approach, and use 64-bin HSV-space color histogram and 80-dimensional edge histogram, which leave us with 144-dimensional raw features per image.

Audio. Audio is recorded with a 11,025 Hz sampling rate in 16 bits PCM format. Various audio features have been used for activity recognition. Lester et al [25] used linear and log-scale FFT frequency coefficients, cepstral coefficients, spectral entropy, band-pass filter coefficients, correlations, integrals, means, and variances. Pärkkä et al [32] used a speech/music discriminator [33] to detect speech from the audio. Kim et al [20] used 26-dim MFCC features along with zero crossing rate, linear predictive coding, volume standard deviation, non-silence ratio, and spectral centroid and pitch, which are a subset of MPEG-7 audio descriptors [29]. Here we use MFCC with 20 filter banks and 13 cepstral coefficients with a 25 ms time window.

Although spectral features such as MFCC are popular in audio processing, they contain enough information to partially reconstruct the speech contents, and cannot be considered as privacy-protecting. One solution to the problem is to use summary features from further processing, such as spectral entropy or energy [47]. In [8], these features were computed on-the-fly on a dedicated device. Our solution to the privacy issue is to use a sparse, short-duration sampling of audio. We sample 250 ms audio fragments for every 5 seconds, effectively discarding 95% of the data before we compute features. The 250 ms fragment, which is shorter than the typical duration of a word, and the sparse sampling together make the overall speech unreconstructible. To get the final feature, we stack MFCC coefficients from non-overlapping windows, which results in 126-dimensional raw features for each audio fragment.

2.3 User Segmentation and Annotation

Through a visual user interface, a user reviews his or her visual log of daily recordings, segments each day into a few ($\sim$10) meaningful events, and annotates each event with

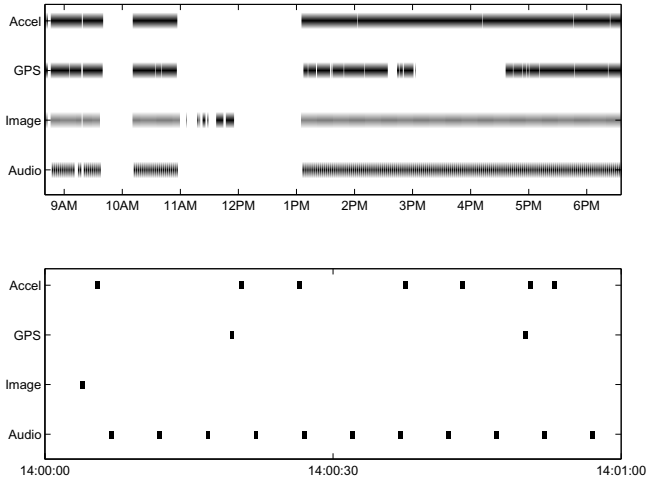


Fig. 1. The figure exemplifies several difficulties of handling heterogeneous multisensory data. (Top) Arrival times of multisensory streams for one day. The figure shows chunks of data are missing for all or some of the sensors. (Bottom) A one-minute portion of the streams around 2 PM, which shows asynchronous arrival times from different sensors. Also images arrive at one sample per minute, while accelerometer and audio samples arrive in higher and more irregular rates.

multiple tags. We use high-level tags that can cover a wide range of daily events, categorized into activity, place, and people (see Table 1). The tags are not mutually exclusive, e.g., a user can select [“eating” and “talking”] in a [“restaurant”] with [“friend” and “family”], or any combination of tags that best describe the event. In our experience, manual annotation of one day by segmentation followed by tagging can be performed in about ten minutes per day, which makes it practical to collect long-term annotated data from non-expert users.

3 Multisensory Bag-of-Words

Processing multisensory data streams in real-life recordings from a smartphone poses several difficulties: 1) samples of different modalities arrive at different times with different rates, e.g., $\sim$1/60 Hz for images vs $\sim$12 Hz for audio fragments; 2) sampling rates of a single modality can change over time, e.g., when other processes on the phone are taking up the CPU resources; 3) chunks of data can be randomly missing for some duration, e.g., when the GPS fails to lock-in indoors, or the camera view is obstructed inside a pocket. Figure 1 shows an example of a day’s recording demonstrating the problems.

These problems can be efficiently handled by the bag-of-words representation. The bag-of-words model assumes that a datum(=document) is a bag of unordered symbols (=words), and that the information about the data is contained fully in the frequency of symbols. Our motivation for the sensory bag-of-words representation is explained by

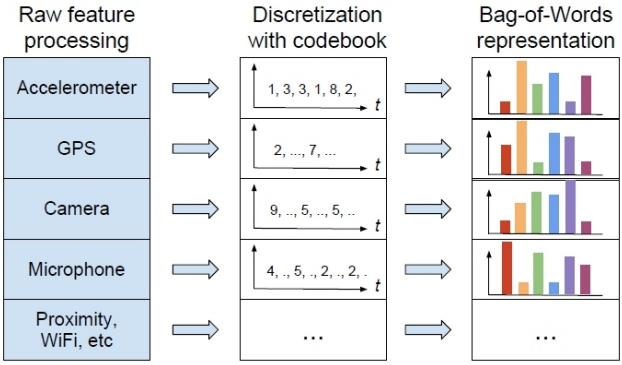


Fig. 2. Multisensory bag-of-words feature representation. Multisensory streams from a smartphone are processed into raw features, quantized into symbolic sequences, and represented as the intermediate bag-of-words features for a given time window.

the following example: during an event in which a person is "eating lunch at a restaurant", it is unrealistic to assume the existence of a single low-level "eating feature" or a "restaurant feature" that can uniquely and reliably identify eating or restaurant. Rather, it is more reasonable to assume that "eating" is a unique *distribution* of multiple low-level features such as walking to the place, standing in line, sitting at a table, and moving upper body; similarly, "restaurant" is also a distribution of low-level features such as a certain location, ambient sounds, and images of people and tables. Therefore, the distribution(=frequency) of low-level features in the bag-of-words representation serves as an intermediate feature that links the raw sensory features with high-level concepts of events. There is also an ample evidence of the utility of the bag-of-words model for activity recognition. Huynh et al. [17] used the bag-of-words representation of accelerometery with a probabilistic topic model, where the frequency of sensory symbols in a 30 minute window was viewed as a document. The authors used the model to classify daily routines such as {dinner, commuting, lunch, office work}. Chennuru et al. [7] introduced an analogy between natural language and 'activity language', and used n-gram statistics of the symbol sequences as intermediate features. The approach was demonstrated with accelerometer data in a classification problem of three activities {walking, running, and cycling}. Wu et al. [44] further extended the approach to symbol-level fusion of GPS and accelerometer, and applied it to classification of 13 activities from a 5 day recording.

In this work, we further promote the multisensory bag-of-word representation as a general framework for transforming any number of heterogeneous sensory streams into homogeneous and normalized features at any desired times scale, that can be combined with any subsequent classification stage (see Figure 2 for illustration.)

To use the model, we first build sensor-specific codebooks (=vocabularies) of K words from the unlabeled collection of data in the training phase. We use K-means clustering to find K dominant modes of low-level features. In the testing phase, raw features are discretized to a symbolic sequence using the codebook of K words. We used

$K = 50$ for all modalities, but it is possible for different modalities to have different values of K. We create one bag-of-words feature for every minute – which we will call a frame – which is the sampling rate of the slowest modality (=image). Missing data can be handled by two ways – by imputing the missing portion from a modality with unbiased prior values (=mean histograms), or by assigning special 'missing' classes to the missing portion of data. Here we use the former approach.

4 Algorithms

We consider automatic annotation as the problem of predicting the presence or absence of each tag at each time frame. The presence or absence of a tag such as an activity or a location likely to be persistent in time rather than changing for every minute. This motivates the use of temporal models for capturing the temporal dependency of nearby states and treating the prediction problem as a sequence labeling problem. In this section, we review several state-of-the-art classifiers that we compare during evaluation.

4.1 Generative Models

Multinomial Naive Bayes. A Multinomial Naive Bayes (MNB)[30] classifier assumes a generative bag-of-words model. Here we adapt the model to our multisensory case. Assume that the data consist of T data frames $W_{1:T} = \{W_1, ..., W_T\}$ and the corresponding T labels $y_{1:T} = \{y_1, ..., y_T\}$. Each frame W_t consists of a collection of observed words from M sensory vocabularies. In MNB, the probability of observing a word v from m-th sensory modality given the state y is

$$p(w^m = v|y = s) = \frac{\exp \phi^m_{v,s}}{\sum_v \exp \phi^m_{v,s}}.$$

Under the 'naive' assumption, the probability of observing all words from M modalities in frame t is

$$p(W_t|y_t = s) = \prod_{m=1}^{M} \prod_{v=1}^{K^m} \left\{ \frac{\exp \phi^m_{v,s}}{\sum_v \exp \phi^m_{v,s}} \right\}^{N^m_v}, \quad (1)$$

where K^m is the size of vocabulary for modality m, and N^m_v is the count of word v from modality m in the frame. The model is trained by finding the maximum likelihood estimates of the parameters $\phi^m_{v,s}$ and the prior probability of a frame belonging to state s: $p(y_t = s) = \pi_s$. Using $p(W|y)$ and $p(y)$, the classification of a test frame W is performed by calculating the posterior probability of each state given the data dictated by the Bayes' rule

$$p(y = s|W) = \frac{\pi_s \cdot p(W|y = s)}{\sum_W \pi_s \cdot p(W|y = s)},$$

and by selecting the state with the highest probability $\arg\max_s p(y = s|W)$.

Hidden Markov Model. A Hidden Markov Model (HMM) assumes that 1) the hidden label state of a frame is conditionally independent of the state of other labels given the state of the preceding frame (a first-order Markov assumption), and 2) the observed feature at a frame is conditionally independent of other states or observations given the state of the frame. The joint probability of a HMM under these assumptions is

$$p(W_{1:T}, y_{1:T}) = p(y_1) \prod_{t=1}^{T} p(W_t|y_t) \cdot \prod_{t=2}^{T} p(y_t|y_{t-1}).$$

The model is specified by the three probabilities: initial probability of the state $p(y_1)$, probability of transition $p(y_t|y_{t-1})$, and emission probability $p(W_t|y_t)$ which models the probability of observing multisensory words in a frame given its state in Eq. 1. Note that if we assume that the state of a frame is independent of the state of any other frame, then HMM simply reduces to MNB.

The parameters of HMM are also learned from maximum likelihood estimation. When the hidden states of the training data is known, as in our case, the optimal parameters of the model can be computed in a closed form. In testing, the most likely sequence of hidden states $y_{1:t}$ of a new sequence $x_{1:t}$, is efficiently found by Viterbi decoding ([35]).

4.2 Discriminative Models 1

Logistic Regression. A Logistic Regression (LogReg) is a log-linear model for discriminative classification of vector data. Let the data consist of T real-valued feature vectors $x_{1:T} = \{x_1, ..., x_T\}$ and T labels $y_{1:T} = \{y_1, ..., y_T\}$, where each feature vector is a concatenated normalized histogram of multisensory words of the frame:

$$x_t = \left[\frac{N_1^1}{N^1}, ..., \frac{N_{K^1}^1}{N^1}, \ ... \ , \frac{N_1^M}{N^M}, ..., \frac{N_{K^M}^M}{N^M} \right]', \tag{2}$$

where K^M and N_v^m are defined the same as in Eq. 1, and N^M is the count of all words for modality m in the frame. Using this feature, the probability of the frame x_t belonging to state s is $p(y_t = s|x_t) = \exp(w_s' x_t) / \sum_s \exp(w_s' x_t)$, where w_s is the vector parameter for state s. For a two-class problem, only one vector w_1 is needed. Training involves learning the parameter w_s that maximizes the conditional log-likelihood of the T training data:

$$\log p(y_{1:T}|x_{1:T}) = \sum_t w_s' x_t - \sum_t \log \sum_s \exp(w_s' x_t).$$

Classification of a test data x is performed by evaluating the conditional likelihoods and selecting the most likely state: $\arg\max_s p(y = s|x)$.

Conditional Random Field. A Conditional Random Field (CRF)[21] is also a log-linear model for discriminative learning of arbitrary undirected graphical models, which was originally proposed to overcome the label-bias problem of HMMs for sequence

labeling. In a (linear chain) CRF, the conditional probability of a state sequence $y_{1:T}$ given a data sequence $x_{1:T}$ is

$$p(y_{1:T}|x_{1:T}) = \frac{1}{Z(x_{1:T})} \exp(\sum_{j,t} w_j f_j(y_t, y_{t-1}, x_{1:T}))$$

where $Z(\cdot)$ is the partition function $Z(x_{1:T}) = \sum_{y_{1:T}} \exp(\sum_{j,t} w_j f_j(y_t, y_{t-1}, x_{1:T}))$.

Note that $f_j(y_t, y_{t-1}, x_{1:T})$ generalizes the emission probability $\log p(x_t|y_t)$ and the transition probability $\log p(y_t|y_{t-1})$ from an HMM, in that it need not be a normalized probability, and that it can use data from all frames $x_{1:T}$ and not just the features of the same frame x_t. In the case where the features depend only on the state and the data of the same frame, that is, $f_j(y_t, y_{t-1}, x_{1:T}) = f_j(y_t, x_t)$, then CRF is similar to LogReg. We use two types of features: the multisensory histogram (Eq. 2) and the persistence of neighboring labels $I(y_{t-1} = y_t)$, where $I(\cdot)$ is an indication function.

For a linear chain CRF, training involves direct maximization of the conditional likelihood over the parameters w_j. In testing, the most likely sequence of the states $y_{1:t}$ of a new sequence $x_{1:t}$ is efficiently found by either an exact or approximate inference [21].

4.3 Discriminative Models 2

Support Vector Machine. A Support Vector Machine (SVM) is a discriminative, large-margin classifier which finds the separating hyperplane $\{w'x + b = 0\}$ for two classes of data. Assume we have the same T feature vectors $x_{1:T}$ and labels $y_{1:T}$ as in LogReg.

The optimization problem for a (linear, soft margin) SVM is to find a maximum margin solution

$$\min_{w,b,\xi} \ \frac{1}{2} w'w + C \sum_i \xi_i \ \text{ subject to}$$
$$y_i(w'x_i + b) \geq 1 - \xi_i, \ \ \xi_i \geq 0, \ \ \forall i,$$

where C is a parameter that allows one to manipulate the relative importance of increasing the margin versus classifying the training examples correctly, and ξ is the slack variable for allowing non-separability of classes. During training, the vector parameter w is found by the optimization, and C is determined by cross-validation. In testing, the state of the given frame x is predicted by the sign of the decision function $f(x) = w'x + b$. A nonlinear version of the SVM is obtained with a nonlinear mapping of samples $x \mapsto \phi(x)$ via a reproducing kernel.

Hidden Markov Support Vector Machine. A Hidden Markov Support Vector Machine (SVM-HMM) [2] is an example of a large-margin discriminative classifiers of structured data [40], with an HMM-like feature space.

Suppose the data is a collection of features sequences $X = \{\tilde{x}^1, ..., \tilde{x}^N\}$ and the corresponding collection of label sequences $Y = \{\tilde{y}^1, ..., \tilde{y}^N\}$, where a single feature sequence and a single label sequence are $\tilde{x}^i = x^i_{1:T} = \{x^i_1, ..., x^i_T\}$ and $\tilde{y}^i = y^i_{1:T} = \{y^i_1, ..., y^i_T\}$ as before.

The optimization problem for a (linear, soft margin) SVM-HMM is to find a maximum margin solution

$$\min_{w,\xi} \ \frac{1}{2}w'w + C\sum_i \xi_i \ \text{ subject to}$$
$$w'\Phi(\tilde{x}^i, \tilde{y}^i) - w'\Phi(\tilde{x}^i, \tilde{y}) \geq 1 - \xi_i, \quad \xi_i \geq 0, \ \ \forall i, \ \tilde{y} \neq \tilde{y}^i,$$

where $\Phi(\tilde{x}^i, \tilde{y}^i)$ is a joint feature-output vector, and C is again a regularization parameter. In SVM-HMM, the joint feature-output vector captures the dependency between feature and label, and between labels of contiguous frames, such as those used in CRF: the multisensory histogram (Eq. 2) and $I(y_{t-1} = y_t)$.

The training and testing procedures are similar to SVM but they require evaluation of $\arg\max_{\tilde{y}} \ w'\Phi(\tilde{x}, \tilde{y})$, which can be done efficiently with a Viterbi-like algorithm from HMM.

5 Experiments

5.1 Lifelogging Data

We collected lifelogging data from a volunteer, who carried the smartphone during daytime in weekdays and in weekends and provided segmentation and annotation of the data. We discarded segments with only private images or segments with less than 5 images, and finally collected 195 hours or 11721 minutes of continuous multisensory recordings of 42 days. The total number of user-defined segments in 42 days was 390, with an average of 9.3 segments per day. Out of the 30 tags (Table 1) that the user used, several tags occurred too infrequently to train and test classifiers reliably. We therefore collapsed those 11 tags that occurred less than 10 times into "other ..." tags, and used the remaining 19 tags in the experiments.

5.2 Comparison of Algorithms

We compare the classification accuracy of the six algorithms: MNB, LogReg, SVM, HMM, CRF, and SVM-HMM. For all algorithms, nearly the same features are used: multisensory word counts for generative models (MNB, HMM), and normalized multisensory histograms for discriminative models (LogRg, SVM, SVM-HMM). Although discriminative models can use more general features than generative models (e.g., histograms from overlapping time windows), we keep the features the same for comparison purposes. We used the packages LIBSVM [6] and the SVM^{hmm}[18], and in-house libraries for the other algorithms. Additional parameters for MNB and HMM are smoothing (=Dirichlet) hyper-parameters α and β for word emission and state prior parameters, which are set to 1. LogReg and CRF use additional regularization (=Gaussian) hyper-parameters, which we set to $\lambda = 10^{-12}$. SVM and HMM-SVM require selection of the coefficient C, which are chosen by five-fold cross-validation within the training data.

Performance is measured by leave-one-day-out classification accuracy for each tag, that is, a model is trained using 41 days and is used to predict the remaining one day.

Table 2. Per-frame classification accuracy of six algorithm for each tag. (mean and s.d. over 42 days.) Boldface means the best result.

		Non-temporal			Temporal		
	Tag	MNB	LogReg	SVM	HMM	CRF	SVM-HMM
Activity	other activity	0.774±0.177	0.804±0.168	**0.806±0.173**	0.784±0.204	0.769±0.230	0.801±0.225
	walk	0.941±0.058	0.968±0.037	0.969±0.040	0.943±0.060	0.950±0.061	**0.973±0.042**
	drive/inside a vehicle	0.904±0.091	0.966±0.055	0.973±0.049	0.883±0.151	0.972±0.045	**0.977±0.047**
	eat/drink	0.837±0.100	0.896±0.081	0.895±0.082	0.845±0.150	0.857±0.137	**0.921±0.094**
	talk/chat/discuss	0.726±0.106	0.797±0.112	0.798±0.113	0.771±0.139	0.824±0.155	**0.870±0.120**
	chores (cook/clean/laundry/etc)	0.846±0.150	0.981±0.039	0.984±0.039	0.772±0.222	0.947±0.097	**0.985±0.039**
	tend to baby	0.798±0.200	0.946±0.091	0.954±0.089	0.747±0.275	0.893±0.150	**0.955±0.089**
	use a computer	0.798±0.113	0.873±0.099	0.875±0.100	0.826±0.144	0.837±0.164	**0.905±0.108**
	read/write on paper/board	0.782±0.168	0.934±0.121	0.937±0.127	0.703±0.272	0.885±0.147	**0.940±0.129**
Place	other place	0.838±0.138	0.940±0.092	0.941±0.093	0.815±0.177	0.913±0.134	**0.942±0.099**
	my home	0.866±0.105	0.898±0.117	0.898±0.123	0.891±0.115	0.890±0.161	**0.935±0.139**
	my office	0.837±0.097	0.891±0.088	0.894±0.089	0.883±0.127	0.908±0.123	**0.941±0.079**
	classroom/meeting room	0.850±0.103	**0.930±0.081**	0.930±0.083	0.837±0.150	0.903±0.163	0.928±0.104
	other's office	0.834±0.143	0.937±0.104	0.939±0.099	0.777±0.229	0.902±0.137	**0.947±0.089**
	restaurant/cafe	0.929±0.062	0.954±0.053	0.958±0.049	0.976±0.032	0.948±0.093	**0.985±0.025**
	outdoor	0.945±0.053	0.971±0.033	0.972±0.034	0.946±0.061	0.951±0.064	**0.978±0.028**
People	other people	0.685±0.136	0.728±0.131	0.729±0.133	0.732±0.187	0.731±0.178	**0.763±0.175**
	my family	0.778±0.173	0.827±0.199	0.825±0.206	0.768±0.180	**0.840±0.193**	0.835±0.227
	colleague(s)	0.767±0.114	0.812±0.102	0.812±0.100	0.816±0.137	0.867±0.130	**0.894±0.100**
Average		0.828±0.070	0.898±0.072	0.899±0.073	0.827±0.077	0.884±0.063	**0.920±0.063**

This process is repeated for 42 times with a different held-out day. The presence or absence of 19 tags at each frame of the test day is predicted by a binary classifier for each tag instead of a multiclass classifier for all tags, since an arbitrary combination of multiple activities, places, and people can occur concurrently at each time frame.

5.3 Results

The per-frame classification rates of six classifiers are summarized in Table 2. The table shows the following trends:

- Accuracy-wise, MNB $\sim$ HMM $<$ CRF $\sim$ LogReg $\sim$ SVM $<$ SVM-HMM.
- Certain tags, such as "walk" and "drive" in activity, and "restaurant/cafe" and "outdoor" in places, are recognized very accurately (> 0.95) for all algorithms.
- Recognition of people is understandably more difficult than activity and place, since we have not used face-related visual features.

It is interesting to see that the two non-temporal models (MNB, LogReg) are not necessarily worse than their temporal counterparts (HMM, CRF). However, it is seen that discriminative models outperform generative models, which is a common observation across different problem domains, and that SVM-HMM outperforms HMM and CRF by a large margin, similar to the results from smart-home data [45].

Lastly, Figure 3 shows predicted tags from SVM-HMM during the course of a day. The predictions are overall satisfactory except for the middle section of the day.

6 Discussion

In this paper, we present a system for collecting and automatically annotating daily experience from multisensory streams on a smartphone. We use a flexible multisensory

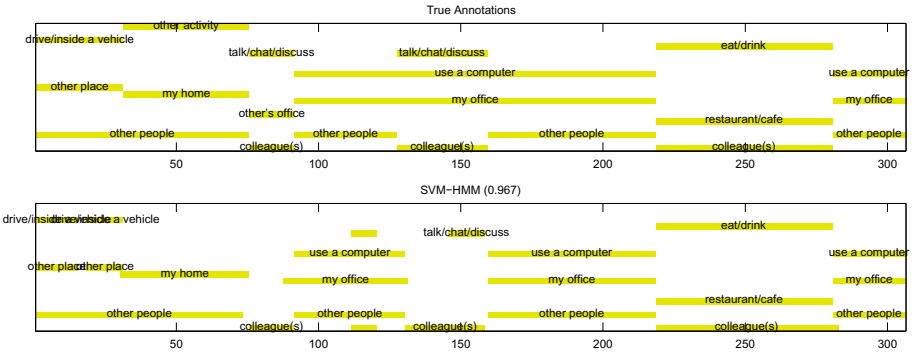


Fig. 3. True vs automatic annotations from data during one sample day. The x-axis is the time in minutes and y-axis is the 19 tags. The yellow dots indicate the positive prediction for each tag at each frame. To reduce visual clutter, we only show tags that last longer than 10 minutes.

bag-of-words representation for incorporating multiple heterogeneous streams, along with state-of-the-art classifiers to predict the tags with real-life data acquired with our system. In the evaluation of algorithms, we compared the performance of various classifiers, and achieved accurate predictions (>0.9) in the majority of tags (15 out of 19) with the SVM-HMM, which outperformed other algorithms by a large margin in almost all tags. The proposed system and algorithms can be immediately used for personal lifelogging applications. For example, using tags as keywords, frames of interest can be retrieved automatically from the vast amount of personal logs. For another example, personal statistics such as the amount of "walking", "driving", or "using a computer" can be calculated from the the predicted labels and provide the user with life-style related feedbacks.

There are several directions to improve the accuracy of automatic annotations with the proposed system. This includes extraction of better features from raw data, and adding more information such as time, Wi-Fi signals, assisted-GPS, ambient light, or proximity. The proposed multisensory framework can elegantly accommodate these inhomogeneous sensory data without modifying the system. The classifiers may also benefit from incorporating a longer-range label dependency or dependency among tags with Factorized HMM [15] or Dynamic CRF [38], as demonstrated in [46]. However, increased model complexity does not always translate to a better performance, and often results in inefficient model learning and inference, in contrast to the models in this paper for which only global optimization is required. While these improvements remain to be explored further, the current work represents a state-of-the-art system in collecting and annotating naturalistic daily experience with multisensory streams from smartphones.

Acknowledgements. The work was supported by the NSF grant IIS-1117707.

References

1. Altun, K., Barshan, B.: Human Activity Recognition Using Inertial/Magnetic Sensor Units. In: Salah, A.A., Gevers, T., Sebe, N., Vinciarelli, A. (eds.) HBU 2010. LNCS, vol. 6219, pp. 38–51. Springer, Heidelberg (2010)
2. Altun, Y., Tsochantaridis, I., Hofmann, T.: Hidden markov support vector machines. In: Proceedings of the Twentieth International Conference on Machine Learning, pp. 3–10. AAAI Press (2003)
3. Bao, L., Intille, S.S.: Activity Recognition from User-Annotated Acceleration Data. In: Ferscha, A., Mattern, F. (eds.) PERVASIVE 2004. LNCS, vol. 3001, pp. 1–17. Springer, Heidelberg (2004)
4. Bieber, G., Voskamp, J., Urban, B.: Activity Recognition for Everyday Life on Mobile Phones. In: Stephanidis, C. (ed.) UAHCI 2009, Part II. LNCS, vol. 5615, pp. 289–296. Springer, Heidelberg (2009)
5. Blighe, M., le Borgne, H., O'Connor, N., Smeaton, A.F., Jones., G.: Exploiting context information to aid landmark detection in sensecam images. In: ECHISE 2006 - 2nd International Workshop on Exploiting Context Histories in Smart Environments - Infrastructures and Design, Ubicomp (2006)
6. Chang, C.-C., Lin, C.-J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27 (2011)
7. Chennuru, S.K., Chen, P.-W., Zhu, J., Zhang, Y.: Mobile lifelogger - recording, indexing, and understanding a mobile user's life. In: Proceedings of the Second International Conference on Mobile Computing, Applications, and Services, Santa Clara, CA, USA (2010)
8. Choudhury, T., Borriello, G., Consolvo, S., Haehnel, D., Harrison, B., Hemingway, B., Hightower, J., Klasnja, P., Koscher, K., LaMarca, A., Landay, J.A., LeGrand, L., Lester, J., Rahimi, A., Rea, A., Wyatt, D.: The mobile sensing platform: An embedded activity recognition system. IEEE Pervasive Computing 7, 32–41 (2008)
9. Chung, P.C., Liu, C.-D.: A daily behavior enabled hidden markov model for human behavior understanding. Pattern Recogn. 41(5), 1589–1597 (2008)
10. Doherty, A.R., Caprani, N., Conaire, C., Kalnikaite, V., Gurrin, C., Smeaton, A.F., O'Connor, N.E.: Passively recognising human activities through lifelogging. Comput. Hum. Behav. 27(5), 1948–1958 (2011)
11. Doherty, A.R., Ó Conaire, C., Blighe, M., Smeaton, A.F., O'Connor, N.E.: Combining image descriptors to effectively retrieve events from visual lifelogs. In: Proceedings of the 1st ACM International Conference on Multimedia Information Retrieval, pp. 10–17. ACM, New York (2008)
12. Farrahi, K., Gatica-Perez, D.: Discovering routines from large-scale human locations using probabilistic topic models. ACM Trans. Intell. Syst. Technol. 2(1), 3:1–3:27 (2011)
13. Fei-Fei, L., Perona, P.: A bayesian hierarchical model for learning natural scene categories. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 524–531 (2005)
14. Gemmell, J., Bell, G., Leuder, R., Drucker, S., Wong, C.: MyLifeBits: Fulfilling the Memex Vision. In: Proceedings of Multimedia 2002 (2002)
15. Ghahramani, Z., Jordan, M.I.: Factorial hidden markov models. Mach. Learn. 29(2-3), 245–273 (1997)
16. Gu, T., Wang, L., Wu, Z., Tao, X., Lu, J.: A pattern mining approach to sensor-based human activity recognition. IEEE Trans. on Knowl. and Data Eng. 23, 1359–1372 (2011)
17. Huynh, T., Fritz, M., Schiele, B.: Discovery of activity patterns using topic models. In: Proceedings of the 10th International Conference on Ubiquitous Computing, pp. 10–19. ACM, New York (2008)

18. Joachims, T.: SVM-HMM : Sequence tagging with structural support vector machines (2008)
19. Kim, E., Helal, S.: Modeling Human Activity Semantics for Improved Recognition Performance. In: Hsu, C.-H., Yang, L.T., Ma, J., Zhu, C. (eds.) UIC 2011. LNCS, vol. 6905, pp. 514–528. Springer, Heidelberg (2011)
20. Kim, I.-J., Ahn, S.C., Ko, H., Kim, H.G.: Automatic lifelog media annotation based on heterogeneous sensor fusion. In: Proceedings of IEEE International Conference on Multi Sensor Fusion and Integration for Intelligent Systems, Seoul, Korea, August 20-22 (2008)
21. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning, pp. 282–289. Morgan Kaufmann Publishers Inc., San Francisco (2001)
22. Lane, N.D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., Campbell, A.T.: A survey of mobile phone sensing. Comm. Mag. 48, 140–150 (2010)
23. Lara, O.D., Labrador, M.A.: A survey on human activity recognition using wearable sensors. Submitted to IEEE Communications Surveys and Tutorials (2012)
24. Lara, O.D., Perez, A.J., Labrador, M.A., Posada, J.D.: Centinela: A human activity recognition system based on acceleration and vital sign data. In: Pervasive and Mobile Computing (2011)
25. Lester, J., Choudhury, T., Borriello, G.: A Practical Approach to Recognizing Physical Activities. In: Fishkin, K.P., Schiele, B., Nixon, P., Quigley, A. (eds.) PERVASIVE 2006. LNCS, vol. 3968, pp. 1–16. Springer, Heidelberg (2006)
26. Li, M., Rozgić, V., Thatte, G., Lee, S., Emken, A., Annavaram, M., Mitra, U., Spruijt-Metz, D., Narayanan, S.: Multimodal physical activity recognition by fusing temporal and cepstral information. IEEE Transactions on Neural Systems and Rehabilitation Engineering 18(4), 369–380 (2010)
27. Logan, B., Healey, J., Philipose, M., Tapia, E.M., Intille, S.: A Long-Term Evaluation of Sensing Modalities for Activity Recognition. In: Krumm, J., Abowd, G.D., Seneviratne, A., Strang, T. (eds.) UbiComp 2007. LNCS, vol. 4717, pp. 483–500. Springer, Heidelberg (2007)
28. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision 60(2), 91–110 (2004)
29. Manjunath, B.S., Salembier, P., Sikora, T. (eds.): Introduction to MPEG-7: Multimedia Content Description Language. Wiley (April 2002)
30. McCallum, A., Nigam, K.: A comparison of event models for naive bayes text classification, pp. 41–48. AAAI Press (1998)
31. Nazerfard, E., Das, B., Holder, L.B., Cook, D.J.: Conditional random fields for activity recognition in smart environments. In: Proceedings of the 1st ACM International Health Informatics Symposium, pp. 282–286. ACM, New York (2010)
32. Pärkkä, J., Ermes, M., Korpipää, P., Mäntyjärvi, J., Peltola, J., Korhonen, I.: Activity classification using realistic data from wearable sensors. IEEE Transactions on Information Technology in Biomedicine 10(1), 119–128 (2006)
33. Penttilä, J., Peltola, J., Seppänen, T.: A speech/music discriminator-based audio browser with a degree of certanity measure. In: Proc. Infotech Oulu Int. Workshop Information Retrieval, pp. 125–131 (2001)
34. Pijl, M., van de Par, S., Shan, C.: An event-based approach to multi-modal activity modeling and recognition. In: Eigth Annual IEEE International Conference on Pervasive Computing and Communications, PerCom 2010, Mannheim, Germany, March 29 - April 2, pp. 98–106 (2010)
35. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)
36. Riboni, D., Bettini, C.: Cosar: hybrid reasoning for context-aware activity recognition. Personal and Ubiquitous Computing 15, 271–289 (2011)

37. Sminchisescu, C., Kanaujia, A., Li, Z., Metaxas, D.: Conditional random fields for contextual human motion recognition. In: Proceedings of the Tenth IEEE International Conference on Computer Vision, vol. 2, pp. 1808–1815. IEEE Computer Society, Washington, DC (2005)
38. Sutton, C., McCallum, A., Rohanimanesh, K.: Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. J. Mach. Learn. Res. 8, 693–723 (2007)
39. Takata, K., Ma, J., Apduhan, B.O., Huang, R., Jin, Q.: Modeling and analyzing individual's daily activities using lifelog. In: Proceedings of the 2008 International Conference on Embedded Software and Systems, pp. 503–510. IEEE Computer Society, Washington, DC (2008)
40. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. J. Mach. Learn. Res. 6, 1453–1484 (2005)
41. Vail, D.L., Veloso, M.M., Lafferty, J.D.: Conditional random fields for activity recognition. In: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 235:1–235:8. ACM, New York (2007)
42. van Kasteren, T., Noulas, A., Englebienne, G., Kröse, B.: Accurate activity recognition in a home setting. In: Proceedings of the 10th International Conference on Ubiquitous Computing, pp. 1–9. ACM, New York (2008)
43. Wang, H., Huang, M., Zhu, X.: A generative probabilistic model for multi-label classification. In: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, pp. 628–637. IEEE Computer Society, Washington, DC (2008)
44. Wu, P., Peng, H.-K., Zhu, J., Zhang, Y.: SensCare: Semi-automatic Activity Summarization System for Elderly Care. In: Zhang, J.Y., Wilkiewicz, J., Nahapetian, A. (eds.) MobiCASE 2011. LNICST, vol. 95, pp. 1–19. Springer, Heidelberg (2012)
45. Wu, T.Y., Hsu, J.Y.J., Chiang, Y.T.: Continuous recognition of daily activities from multiple heterogeneous sensors. In: AAAI Spring Symposium: Human Behavior Modeling, pp. 80–85. AAAI (2009)
46. Wu, T.-Y., Lian, C.-C., Hsu, J.-Y.: Joint Recognition of Multiple Concurrent Activities using Factorial Conditional Random Fields. In: AAAI Workshop on Plan, Activity, and Intent Recognition, Technical Report WS-07-09. The AAAI Press, Menlo Park (2007)
47. Wyatt, D., Choudhury, T., Kautz, H.: Capturing spontaneous conversation and social dynamics: A privacy sensitive data collection effort. In: Proc. of ICASSP (2007)
48. Zappi, P., Stiefmeier, T., Farella, E., Roggen, D., Benini, L., Tröster, G.: Activity recognition from on-body sensors by classifier fusion: Sensor scalability and robustness. In: 3rd Int. Conf. on Intelligent Sensors, Sensor Networks, and Information Processing, pp. 281–286 (2007)
49. Zhu, Y., Arase, Y., Xie, X., Yang, Q.: Bayesian nonparametric modeling of user activities. In: Proceedings of the 2011 International Workshop on Trajectory Data Mining and Analysis, pp. 1–4. ACM, New York (2011)

Personalized Energy Consumption Modeling on Smartphones

Yifei Jiang[1], Abhishek Jaiantilal[1], Xin Pan[1], Mohammad A.A.H. Al-Mutawa[1], Shivakant Mishra[1], and Larry Shi[2]

[1] University of Colorado at Boulder, CO 80309, U.S.A.
{yifei.jiang,abhishek.jaiantilal,shivakaht.mishra}@colorado.edu
[2] University of Houston, TX 77204, U.S.A.
larryshi@cs.uh.edu

Abstract. Energy has emerged as a key limitation in smartphone usage. As a result, optimizing power consumption has become a key design issue in building services and applications for smartphones. Understanding user behavior and its impact on energy consumption of smartphones is a key step for addressing this problem. This paper provides an in-depth study of user behavior and energy consumption of smartphones by analyzing smartphone data collected from twenty smartphone users over a period of three months. In particular, correlations between power consumption and factors such as time of day, user's location, remaining battery power, recent phone usage history, and phone's idle and active states have been studied. The results show varied levels of correlations between a user's phone usage and these factors, and can be used to model and predict smartphone power consumption.

Keywords: mobile computing, power usage, mobile power usage, user study.

1 Introduction

Smartphones are becoming an amalgam of multiple services, including online, social, video, music, gaming, payments, location-based services, and augmented reality. Because of such multifarious functionalities and applications, energy consumption of smartphones is rapidly increasing and becoming highly variant. For example, a typical smartphone features around five hours talking time on 3G and 150 hours standby time. However, these figures drop sharply when users play games, surf the web, or use GPS navigation. Because of the high and variant energy usage, battery lifetime of smartphones is getting shorter and much more unpredictable. As a result, better energy management techniques for smartphones are needed.

An understanding of energy consumption of smartphones as well as user behavior that impacts this energy consumption is a key requirement for designing effective energy management techniques. For example, if we can predict that a user will be running some power-hungry application on their smartphone over

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 343–354, 2013.

the next two hours and they will be at a place where there is no source for charging the phone, the system can then recommend the user to charge their phone immediately. Another example: power-aware scheduling mechanisms can be designed to pause non-urgent applications and services if we can predict that the user will be running some power-hungry application soon. Finally, knowledge of a smartphone's power consumption behavior can aid in detecting abnormal conditions (like a malware is running on the phone).

The main goal of this paper is to study the relationship between user behavior, their environment and energy consumption. Our study is motivated by several important observations. First, user location plays an important roles in phone usage. For example, users may surf the web or read emails when staying at an airport, but they may play a game when they are at home. Second, the amount of battery power remaining in the phone impacts user behavior. For example, a user is less likely to play (power-hungry) games if the remaining battery lifetime is quite low. Finally, different locations have different communication environments due to varying GSM signals and availability of WiFi access points.

This study of user behavior and its impact on energy consumption of smartphones is done by analyzing user behavior and power consumption data collected from twenty smartphone users over a period of three months. In particular, we study correlation between power consumption and factors such as time of day, user's location, remaining battery power, recent phone usage history, and phone's idle and active states. The results show varied levels of correlation between a user's phone usage and these factors, and this information can be used to model power consumption of a user's smartphone.

The rest of the paper is organized as follows. Section 2 briefly discusses the important related work. Section 3 describes our data collection methodology. Section 4 presents the data we have collected. Four factors that impact energy consumption of smartphones have been identified: time of day, user location, remaining battery life, and user's recent phone usage history. An analysis reveals that energy consumption pattern varies based on when a phone is idle and when it is active. Section 5 provides this analysis. Finally, Section 6 provides a discussion and concludes the paper.

2 Related Work

Earlier research has shown that battery consumption and charging vary in context [14,4,2,11,12]. Trestian *et al.* [14] examined the correlation between location and application usage from 280,000 clients of a 3G mobile network and found that user interactions with mobile phone vary according to users' location. Similarly, Banerjee *et al.* [2,11,12] studied battery charging behaviors on mobile systems and noted that most charging behaviors were driven by context, including location and time of day. Falaki *et al.* [4] conducted a study of user interactions with mobile phones and examined the corresponding energy drain rates. They found that considerable variation among user activities resulted in diverse energy consumption patterns, which indicates that learning and adapting

Table 1. Data collection on smartphone

Data	Sample rate
WiFi fingerprint	Every 10 minutes
Battery current	Every 5 seconds
Remaining battery	Call back
GPS	Call back
CPU load	Every 1.5 minutes
Screen status	Call back
Phone calls	Call back
Time stamp	For each data log

to personalized user behavior is likely to be more effective for improved mobile energy management.

In a different context, extensive research has also been done towards understanding power management at the hardware level. Simunic *et al.* [13] provided system-level dynamic power management algorithms to schedule idle components into lower power states automatically. Carroll *et al.* [3] conducted exhaustive measurements and analysis on different components of mobile devices.

There are in general two categories of location sensing algorithms – geometry and fingerprint. Geometry based algorithms [1,6] cluster geo-coordinates that belong to the same meaningful location into the same cluster. The location is recognized by checking whether the device's current geo-location falls into the geometric shape of any known location. Assuming that the radio fingerprints of locations are rather stable and unique, fingerprint based approach [5,9,15,7,8] can tell whether two locations are close or far apart according to the similarity between two fingerprints. SensLoc [8] uses accelerometers to detect movement and turns off GPS/WiFi sampling when users are stationary in order to save energy, thereby making the location sensing more energy efficient. Besides location sensing, iLoc [10] provides a predictive model to forecast future location-state transitions.

3 Data Collection

We developed a data collection application for two smartphone platforms: Nokia N900 smartphone[1] running Maemo Linux and smartphones running Android[2]. The application is developed for monitoring user's power usage and their context information such as location information, battery status and time. Furthermore, the usage information of each hardware component, e.g., GPS, screen, and CPU, are also collected by the application. The data and sample rate we used are listed in table 1. The energy overhead is less than 5 mw.

[1] `http://ml.cs.colorado.edu/~abhi/mobien/`
[2] `https://market.android.com/details?id=edu.colorado.mobien`

To determine location, we used information from both the cellular sites and the WiFi access points [8]. The cellular information (GSM Cell ID and CDMA Base station ID) provide us with a coarse location, while the WiFi scan information provide us with a finer location. Note that we do not identify the actual geographical location of the smartphone (or the user), as we are mostly interested in differentiating among the different locations that the user visited. We did not use GPS to determine location because it consumes more power and does not work inside buildings.

Over a period of three months, we collected data from twenty users who installed our application. Ten of these users are known to us and they installed our application on our request, while others installed our application by picking up from the Android Marketplace. There were six Nokia N900 user and fourteen Android users. Among the users known to us, some are graduate students and others work in industries. Of the twenty users, data collected from six users (three Android and three Nokia users) was most complete, i.e. it spanned the entire three-month period. We have chosen the data collected from these six users for our detailed analysis reported in this paper. The data collected from the remaining fourteen users spanned from two weeks to less than three months. All observations that we report in this paper do hold for these data as well. As we are primarily interested in usage patterns of users with reference to time and location, we have excluded portions of the data collected when the phone was charging and when a user was traveling between locations. The charging and discharging behavior was caught by a system API of the OS platforms. We were able to distinguish whether the user was stationary or traveling between locations by observing the change in the Wifi access points.

4 Phone Usage Analysis

The overall energy consumption of a smartphone includes the energy consumed during the phone's *active* as well as *idle* states. In active state, smartphones provide one or multiple types of information to user, including screen output, audio ouput and vibration ouput; and in idle state, smartphones do not provide any information to user, but the device may run backstage process such as GSM communication and customized data collection application. In the active state, a user is running some application and using the phone. Energy consumed during the active state is on average ten times higher than the energy consumed during the idle state. Thus active state is mostly responsible for the overall energy consumption. After a detailed analysis of the power consumption data collected, we have identified four factors that impact smartphone usage. These factors are: time of day, user location, remaining battery level, and recent phone usage history. We first analyze these factors, and then in the next subsection we explore how these factors affect energy consumption in active and idle states of the phone.

We first look at the percentage of the time that phone is in use across 20 users in our dataset. As showed in Figure 1, the y-axis shows the fraction of phone

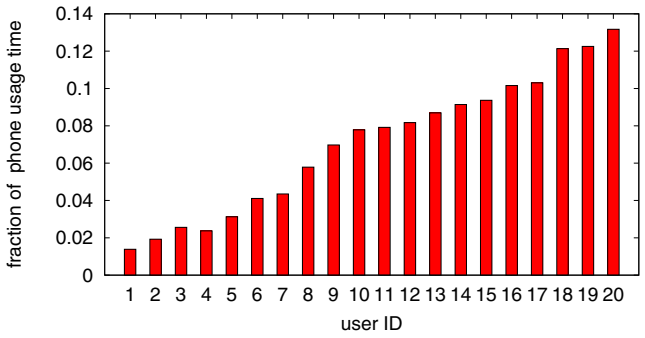


Fig. 1. Correlation between Time and Phone usage

usage time and x-aixs is the different user IDs sorted by phone usage percentage. The plot shows that 1) most of the users did use their phone actively, and on average they used their for 90 minutes per day. 2) the phone usage profile are very different across different users.

4.1 Time of Day

Logically, phone usage is related to users' daily schedule and routine activities and thus likely to be highly correlated with time of day. For example, a user may always sleep around 11 PM and get up at around 7 AM. Therefore, his/her phone is likely to be, everyday, in an idle state from 11 PM to 7 AM. On the other hand, the phone usage is typically high during some routine activities, e.g. during lunch time, users may often use their phones to check email.

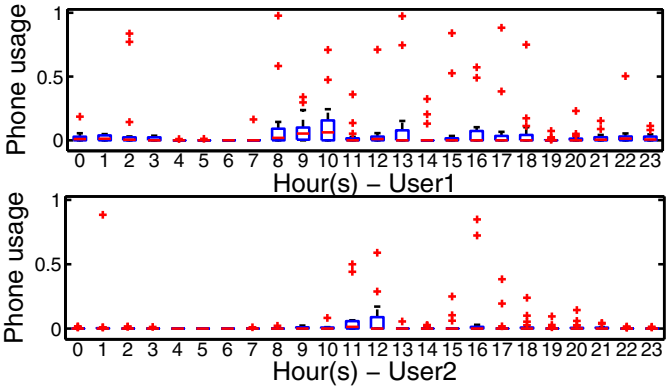


Fig. 2. Correlation between Time and Phone usage

Figure 2 shows the box-plot of phone usage over the 24 hours of a day for two users in our three month experiment. The phone usage is calculated as a percentage of the phone's active time during an hour. Red plus signs in this plot show the outliers. For example, for user 1 in hour 2, the percentage of time that the phone is in active state is below 3% all the time during the three-month period except for the three instances that are shown as outliers. An important observation we made from this figure is that the usage pattern for a user is generally quite uniform (smaller-sized box with low number of outliers). Although, this figure presents the data collected from only two users, we have also observed similar behaviors from all the twenty users based on the data we collected. This implies that a user's phone usage does not vary significantly everyday (over different hours). Thus, it is possible to model a user's smartphone power consumption based on the time of day and also make fairly accurate prediction about how much power will be consumed in the future. Of course, the presence of some, albeit small number of outliers as well as large-sized boxes for certain hours indicates that there is a possibility of error in this prediction especially when the prediction is made for those hours.

Also, Figure 2 shows that the phone usage of different users follow different patterns over time. For example, user 1 has high phone usage between 8 AM to 10 AM, while user 2 has high phone usage between 11 AM to 12 PM.

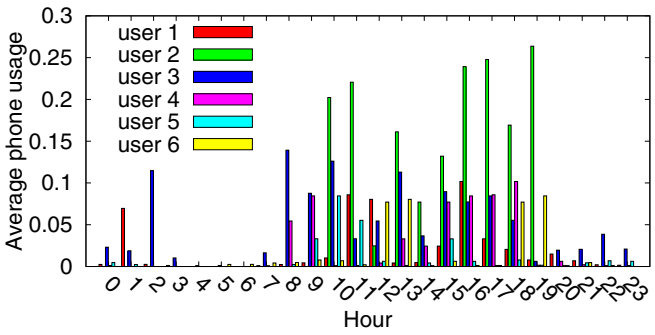


Fig. 3. Correlation between Time and Phone usage

Figure 3 shows this difference more clearly. Here we depict the average phone usage of six users over 24 hours. This figure indicates that different users have different usage pattern over time. Therefore, a personalized analysis is required.

4.2 Location

Location is another important factor that affects a user's phone usage because locations are typically indicative of the user's activities and associated phone

usage. Figure 4 shows the box-plot of phone usage at different locations. This figure includes the data collected from six users. Note that we consider the same physical location visited by two different users as two separate locations in this figure because different users may have different behaviors at the same location. The phone usage is computed for each visit to the location.

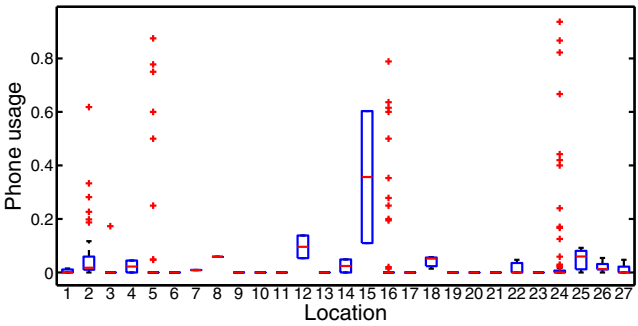


Fig. 4. Correlation between Location and Phone usage

We identify three types of location-based patterns when users visit different locations. First location-based usage pattern: phone usage is almost constant (smaller-sized box) for each visit. The phone usage at locations 4, 6-14, 17-23, etc. follow this pattern. This type of pattern happens at the locations where users have similar behaviors for each visit, or users never use their phones at those locations. Second location-based pattern: phone usage is similar for most visits but occasionally deviates from the normal profile (small-sized box and some outliers). The phone usage at locations 2 and 16 follows this pattern. The outliers here are typically caused by emergent or irregular events, such as an infrequent call. Third location-based pattern: phone usage varies a lot over different visits to the same location. The phone usage at location 15 follows this pattern. Thus, a power consumption model based on a user's locations that considers either the first or the second location-based pattern can make fairly accurate predictions about how much power will be consumed based on where a user is located.

4.3 Time of Day and Location

While both time of day and location correlate quite well with power consumption, we note that there is a possibility of a higher correlation between power consumption and (time of day, location). The intuition behind this is that a user may use his/her phone for different purposes at the same location at different times. For example, a student may check emails during daytime in the university, while play games at night. Figure 5 shows the box-plot of phone usage

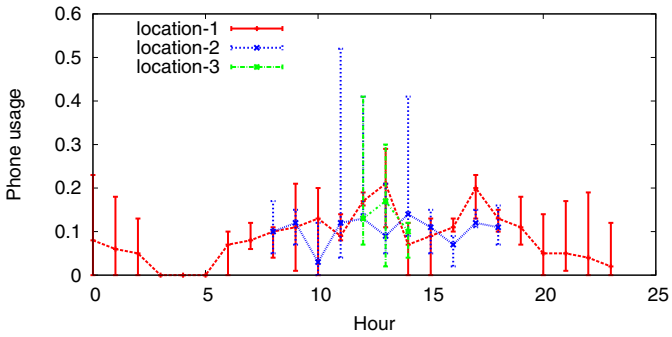


Fig. 5. Correlation between phone usage and (location, time of day)

for one user[3] at different locations and time. The figure shows three locations that the user frequently visited during the study period. Location 1 is most likely the user's residential place. Meanwhile, the user spent significant amount of time during day time at location 2 and location 3. Comparing location 2 with location 3, location 3 has a much narrow visit window. Figure 5 does not show locations with stay time less than one hour. According to figure 5, the user visited certain locations during predictable hours of a time (e.g., location 3). Figure 5 suggests that phone usage varied during the time when the user was at a particular location. Phone usage was more evenly distributed across time at residential place than other locations. During busy hours of a day (11am to 2pm), there was generally more phone usage and larger variance in usage pattern.

4.4 Battery State

We now consider battery state collected from the current battery level displayed on smartphone UI. In our analysis, we use eight discrete battery states (1-8) to examine the correlation between battery state and phone usage. Battery state 1 indicates that the battery is close to empty, and battery state 8 indicates that the battery is fully charged.

Figure 6 shows the correlation between the battery state and phone usage for six users. As shown in the figure, the average phone usage decreases with a decrease in the remaining battery level. This behavior is uniform across all users; it clearly indicates that a user's behavior in terms of the running applications is affected by the current state of the phone battery. Phone usage is low when the remaining battery level is low because 1) users want to keep the phone alive till the next charging; and 2) the actual remaining battery time is hard to estimate so that users tend to rely on the battery state provided by phone UI. This indicates that remaining battery time can be used to predict user behaviors in terms of if and how much they will use their phones.

[3] Data from the other users show similar trend. However, due to page limit, we cannot incorporate each user's location-time figure.

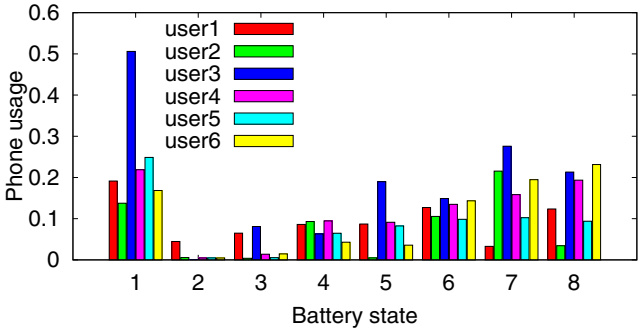


Fig. 6. Battery state vs Phone usage

The main exception here is when the battery state is 1, i.e. when the remaining battery time is extremely low. In that state, phone usage is actually higher than all other states for all six users. We traced this abnormality to the system periodically providing the user an alarm for low-battery, e.g. turning on screen, making a sound, vibrating, etc., which made the system active.

4.5 Recent Phone Usage History

Recent phone usage history may be used to predict a user's current phone usage. This hypothesis is based on our observation that a user's phone usage pattern typically doesn't change abruptly. So, users who have been using their phones quite actively in the recent past will continue to use their phones actively in the near future and vice versa for users who do not use their phones actively.

In order to explore this hypothesis, we analyzed the data collected from consecutive days for each user. If a user has a 30 day history period, we consider 29 pairs of data points for days, i.e (1,2), (2,3), ..., (29,30). For each user and a given average phone usage value on the previous day, we find the average phone usage value for the next day; e.g. for value x=0.1, we find all the tuples for which previous day value was 0.1 and then find the average of the next day's phone usage which we denote by y. We plot the (x,y) value correspondingly on the x/y-axes in Figure 7.

As shown in Figure 7, for most users, if the phone usage on previous day was higher, the phone usage on the current day will be, on average, higher. An interesting observation shown in the figure is the following: if a user has very low phone usage (e.g. < 0.02) on one day, usually, the phone usage would go up significantly on the next day.

5 Energy Consumption Analysis

5.1 Idle State

Energy consumption in idle state is dependent on two factors: programs running in the background and the ambient signal quality. Background programs directly

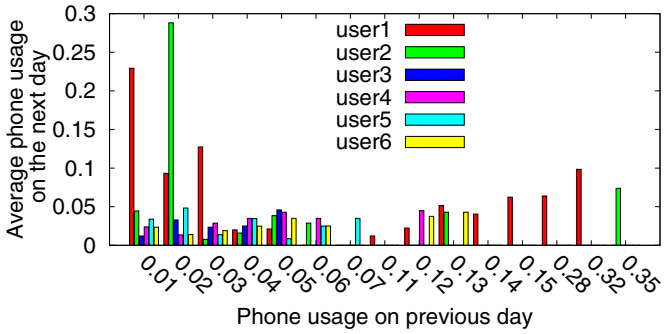


Fig. 7. Recent (Phone usage) history vs Phone usage

determine the usage/activeness of hardware components, e.g. CPU, networking and sensors; whereas, ambient signals affect power consumption of radio components, e.g. poor GSM/CDMA signal strength causes higher energy consumption. In particular, the second factor is related with the smartphone's location. In Figure 8, we plot the correlation between the location and the energy consumption in idle state. As shown in the figure, for most locations, the variability in power consumption is relatively low (small-sized boxes with low number of outliers). This indicates a very strong correlation between the power consumption and the location when the phone is in idle state.

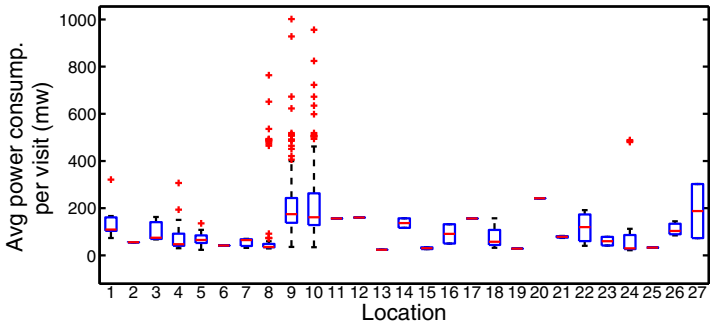


Fig. 8. Location vs Power consumption in Idle state

5.2 Active State

Energy consumption of a smartphone in active state is much higher than the average energy consumption in idle state. In this state, energy consumption depends on the running applications/software and the utilized sensors. The running applications are varied and user-specific. Since location is an important context that affects the running applications on the phone, we explore the correlation between energy consumption in active state and the location.

Figure 9 shows a plot of this correlation. Based on this plot, we can clearly divide user locations into two categories. The first category consists of locations where power consumption is relatively constant (small-sized boxes and low number of outliers). Examples include locations 2-6, 8, 10, and so on; for these locations, the running applications are strongly correlated with the location. The second category consists of locations where power consumption is highly variant (large-sized boxes and/or high number of outliers). Examples include locations 1, 7, 11-13, and so on.

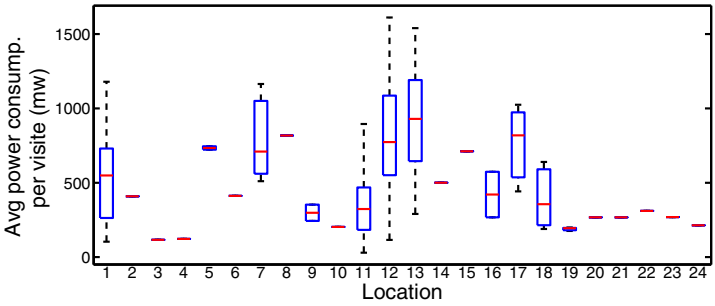


Fig. 9. Location vs Power consumption in Active state

Our analysis of power consumption in active and idle states indicates that a user-centric power consumption model, which incorporates these two states along with time of day, location, remaining battery level, and recent phone usage history, will help to provide more accurate predictions about power consumption.

6 Conclusions

In this paper, we studied and analyzed the interrelationships between smartphone usage behavior, environment, and energy consumption. We have identified four factors — time of day, user location, battery state, and recent phone usage history — that can be used to predict phone usage and energy consumption. We also explored the impact of location on energy consumption under both idle and active phone states. Our study results provide useful insights for modeling smartphone energy consumption, particularly taking into account the context of location.

While we have focused on individual factors and analyzed their impact on energy consumption, it would be interesting to investigate if there exists any stronger correlation between energy consumption and a combination of two or more factors. For example, our results show that there is a high correlation between energy consumption and time of day. Similarly, the results also indicate that there is a high correlation between energy consumption and user's location. We may find an even stronger correlation between energy consumption

and <time of day, location>. As part of continuing efforts, we are developing quantitative energy consumption models based on the data collected from users. Then we are going to integrate these models into power management services present in smartphones. The end result will be an intelligent and energy efficient mobile platform based on energy consumption patterns of individual smartphone user.

References

1. Ashbrook, D., Starner, T.: Using GPS to learn significant locations and predict movement across multiple users. Personal Ubiquitous Computing 7(5), 275–286 (2003)
2. Banerjee, N., Rahmati, A., Corner, M.D., Rollins, S., Zhong, L.: Users and batteries: interactions and adaptive energy management in mobile systems. In: UbiComp 2007 (2007)
3. Carroll, A., Heiser, G.: An analysis of power consumption in a smartphone. In: USENIX ATC (2010)
4. Falaki, H., Mahajan, R., Kandula, S., Lymberopoulos, D., Govindan, R., Estrin, D.: Diversity in smartphone usage. In: MobiSys (2010)
5. Hightower, J., Consolvo, S., LaMarca, A., Smith, I.E., Hughes, J.: Learning and recognizing the places we go. In: ACM Ubicomp, pp. 159–176 (2005)
6. Kang, J.H., Welbourne, W., Stewart, B., Borriello, G.: Extracting places from traces of locations. ACM Mobile Computing Communication Review 9(3) (2005)
7. Kim, D.H., Hightower, J., Govindan, R., Estrin, D.: Discovering semantically meaningful places from pervasive rf-beacons. In: ACM Ubicomp (2009)
8. Kim, D.H., Kim, Y., Estrin, D., Srivastava, M.B.: SensLoc: Sensing everyday places and paths using less energy. In: ACM SenSys (2010)
9. Laasonen, K., Raento, M., Toivonen, H.: Adaptive On-Device Location Recognition. In: Ferscha, A., Mattern, F. (eds.) PERVASIVE 2004. LNCS, vol. 3001, pp. 287–304. Springer, Heidelberg (2004)
10. Ma, Y., Hankins, R., Racz, D.: iloc: a framework for incremental location-state acquisition and prediction based on mobile sensors. In: CIKM (2009)
11. Rahmati, A., Qian, A., Zhong, L.: Understanding human-battery interaction on mobile phones. In: MobileHCI (2007)
12. Rahmati, A., Zhong, L.: Fast track article: Human battery interaction on mobile phones. Pervasive Mob. Comput. 5 (October 2009)
13. Simunic, T., Benini, L., Glynn, P., De Micheli, G.: Dynamic power management for portable systems. In: MobiCom (2000)
14. Trestian, I., Ranjan, S., Kuzmanovic, A., Nucci, A.: Measuring serendipity: connecting people, locations and interests in a mobile 3g network. In: IMC 2009 (2009)
15. Yang, G.: Discovering Significant Places from Mobile Phones – A Mass Market Solution. In: Fuller, R., Koutsoukos, X.D. (eds.) MELT 2009. LNCS, vol. 5801, pp. 34–49. Springer, Heidelberg (2009)

An OPENID Identity Service for Android, Based on USIM Secure Elements

Pascal Urien

Telecom ParisTech.,
23 Avenue d'Italie, 75013, Paris, France
Pascal.Urien@Telecom-ParisTech.fr

Abstract. This paper presents a new identity platform based on the OPENID standard for Android mobile and working with the Open Mobile API, which has been recently defined by the SIM Alliance committee. This innovative service comprises four components, an embedded USIM TLS stack, an Android proxy application (PS), an identity provider (IdP) server, and an OPENID authentication server. USIM modules are issued with a device certificate and may thereafter download other certificates from the IdP server. All HTTP exchanges are secure by the USIM TLS stack. This platform may work with about one million of WEB sites, which are today compatible with the OPENID framework.

Keywords: Security, OPENID, USIM, Android, Secure Element.

1 Introduction

The mobile applications rocket is fuelled by the development of the Android devices sales. According to [1], this open operating system could power about 50% of smartphones by 2015. Android [8] was originally created by the company Android Inc, bought in 2005 by Google. The first mobile (the HTC G1) was commercialized in 2008. Some Android mobile are open, such as the Nexus S, i.e. quite all code sources (except some libraries imported under a binary format) are available in dedicated repositories managed by the GIT and REPO tools [7]. It is therefore possible to compile and load OEM versions for some models. Furthermore, Android works with a UNIX kernel; there are many tools over the WEB enabling to root the system, which consequently may be modified without manufacturers or operators agreement. We recently made a reverse engineering of mobile banking applications, which shows that most of them do not rely on Android libraries, but embed their own cryptographic resources, typically used for TLS sessions with online account servers. A way to solve trust issues in mobile is to run some parts of applications in tamper resistant components, usually referred as Secure Elements (SE). A SE is a secure microcontroller protected by various physical and logical countermeasures. There are four classes of such devices in today mobiles: USIM [11] chips mandatory for 3G/4G/LTE authentication, NFC (Near Field Communication) controllers [14], micro SD cards [9][10] including secure microcontrollers, and extern contactless smartcard feed by NFC reader. In this paper we focus on USIM device because we target an operator service.

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 355–366, 2013.

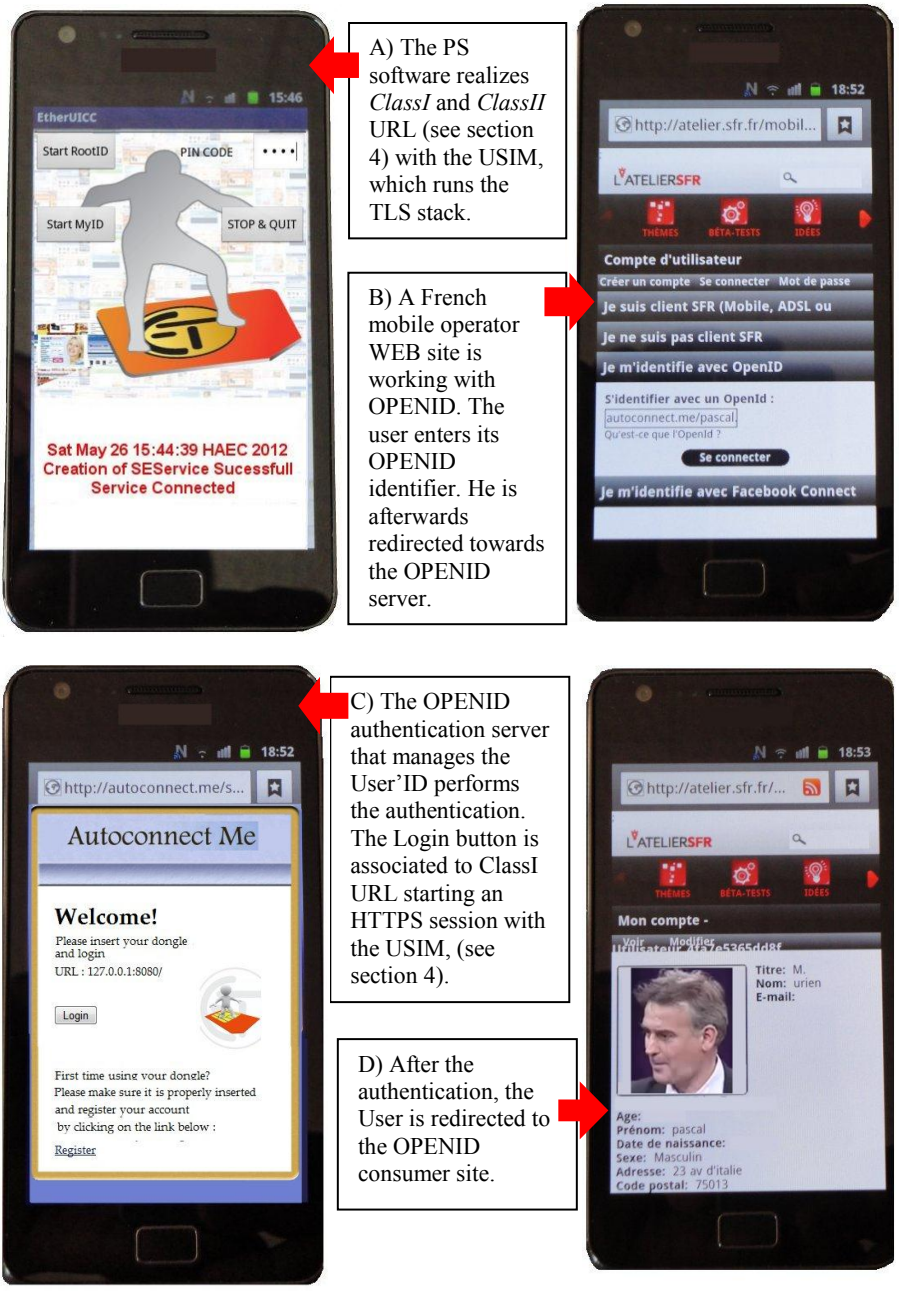


Fig. 1. Illustration of the User's experience OPENID Identity Service

A TLS [18] stack is running in the USIM it performs the booting of TLS sessions, i.e. until the reception of the server finished message for the TLS full mode, and the transmission of the client finished message for the abbreviated (or resume) mode. We use TLS with certificates for both client and server sides, i.e. there is a strong mutual authentication based on PKI between these two entities. The TLS session is afterwards transferred to the android mobile application which collects the CipherSuite and the associated KeysBlock values needed by the record layer for encryption and integrity procedures.

In android, telephony operations (ingoing calls, outgoing calls…) are controlled by the RIL (Radio Layer Interface) software component [6], acting as a TCP daemon. It is used for packets exchange between the application processor (in which is running the operating system) and the baseband processor that handles radio interfaces and controls the USIM. For security reasons the RIL forbids information exchange between the application processor and the USIM. Nerveless the SIM Alliance has defined an Open Mobile API [4] that unifies SE access in mobile environments, and which allows interactions with applications embedded in the USIM. For that purpose the baseband manufacturer delivers a special RIL version that enables communication with the USIM, typically via AT-CSIM [25] commands. Furthermore this API manages an "Access Control Rules Database & Engine", which checks that the application running in the SE (the USIM in our case) authorizes interactions with mobile applications, whose rights and integrity are checked by a certificate. Elements of this security policy are stored according to a PKCS#15 scheme in the USIM file system [5]. In this context we have developed an identity platform based on the OPENID standard [19], which according to [2] is supported by about one million of WEB sites. The basic idea is that USIM are issued with a TLS stack and a device certificate. A mobile proxy application establishes the glue between the browser and two classes of remote servers. An Identity Provider (IdP) server delivers new identities (i.e. certificate and private key) to the USIM. An OPENID authentication server that trusts these identities (i.e. the Certification Authority -CA- issuing the certificates) authenticates the USIM, via TLS sessions with mutual authentication.

The user experience is illustrated by figure 1. The mobile bearer starts the proxy application that establishes the glue between the browser and the USIM. He then works with a WEB site that requires an OPENID identifier for login operation. He is thereafter identified by an HTTPS session with the OPENID authentication server.

Finally he is redirected to the previous site that displays the success of the authentication operation.

This paper is constructed according to the following outline. Section 2 introduces the android architecture, more specially the Nexus S device. Section 3 presents the "Open Mobile API" and its policy access rules for the USIM. Section 4 describes our new identity service that comprises four components, an embedded USIM application, a mobile android application, an identity provider server, and an OPENID authentication server. Finally section 5 concludes this paper.

This work has been done with the collaboration of the EtherTrust Company. It has been partially funded by the STREP Project 288349, SecFuNet.

2 About Android

The Android system [8] is mainly powered by a Linux kernel and a Dalvik virtual machine. The figure 2 illustrates the hardware and software architecture for a Nexus S mobile. There are two processors, an application processor that runs the operating system, a baseband processor that manages all the radio packet exchange. Two secure elements are available, the USIM and the NFC controller.

The Linux kernel manages memories and processes. It also handles the IO drivers (/dev/devices) needed for the access to numerous peripheral devices that are controlled by the application processor. Two low level drivers handle components giving access to Secure Elements: the baseband driver; and the NFC controller driver embedding an optional tamper resistant microcontroller.

The kernel is built with a tool named GIT and is produced under the zImage format. The Android system is usually referred as the ROM, which is made of three parts. 1) The boot image (boot.img), which stores the kernel image. 2) The system image (system.img), which store libraries and applications. 3) The recovery image (revovery.img), a minimum version of the system used to download new ROMs.

The ROM is produced from several logical elements: 1) the kernel imported in binary format; 2) the Dalvik virtual machine code; 3) the libraries code, some of them such as the RIL or NFC components are proprietary and imported in binary format and required a license agreement; 4) a JAVA framework based on JINI interfaces with native libraries; 5) applications offering facilities such as telephony or NFC.

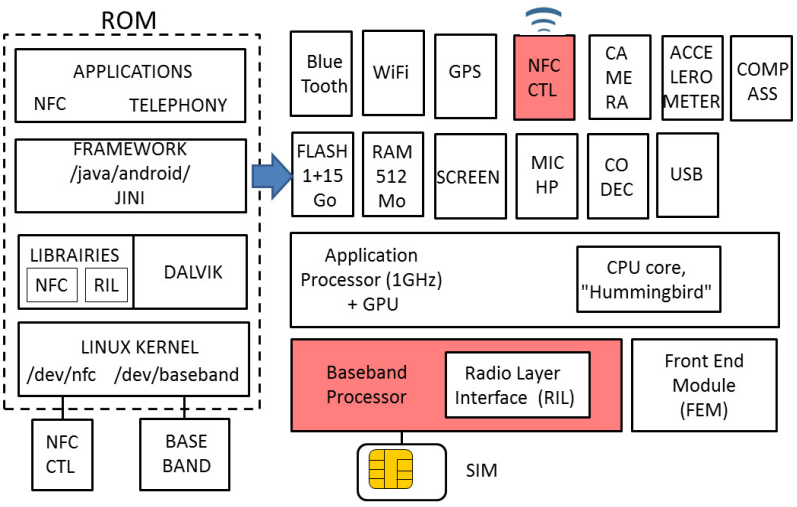


Fig. 2. Android operating system, hardware and software components

A mobile is released with an initial ROM code, which is modified by signed update files (update.zip). Nerveless it is possible to root a mobile phone, i.e. to get root privileges. This allows modifying parts of the ROM stored in the FLASH memory. As illustrated in the previous section some Android platforms are open, i.e. it is possible to build (OEM) ROM from source codes, and to flash it.

For these reasons most of android applications that need trust, work with they own cryptographic resources.

Secure Elements (USIM, NFC Controller, Secure SD...) are processors whose code can't be loaded/modified without specific authorizations. The USIM chip is a tamper resistant device mandatory for 3G/4G and LTE networks. The communication with the application processor is managed by the RIL (Radio Layer Interface) entity, which unfortunately in the current Android model does not provide access to the USIM.

The Open Mobile API detailed in the next section builds a generic framework for the management of secure elements, and works with RILs providing access to the USIM, typically via AT-CSIM commands [25].

3 About the Open Mobile API

3.1 About Secure Elements

Secure elements are tamper resistant devices based on secure microcontrollers. Security is enforced by multiple hardware and software countermeasures. They comprise CPU (8, 16 or 32 bits) with modest computing power, nonvolatile memory (FLASH or E^2PROM, about 100KB), ROM (a few hundred KB) and RAM (about 10KB). Most of them are equipped with a java virtual machine (JVM) and run applications written in javacard [13], a subset of the java language. Applications are identified by an AID (Application Identifier) whose length is 16 bytes at the most.

SE exchange commands, whose maximum size is about 256 bytes, which are called APDUs and defined by the ISO7816 standards [12]. A request APDU comprises a five bytes header, CLA the class of operation (for example 00 for ISO request, A0 for GSM), INS the instruction (B0 for reading, D0 for writing…), two bytes P1 P2, and a P3 parameter either the length of information to be written or the expected length of information to be read.

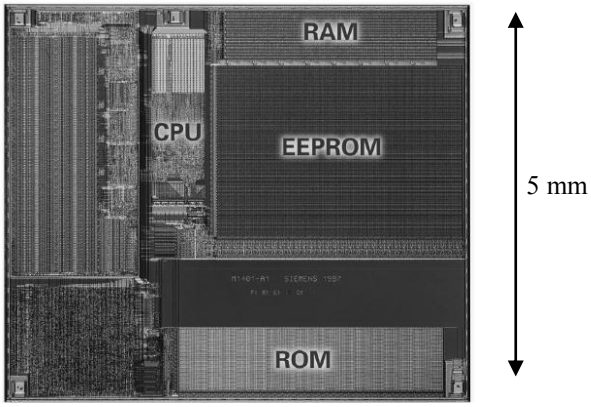


Fig. 3. Illustration of an USIM chip from Siemens ([11])

The Card Manager [15] is an application identified by its AID, which can download and activate software in Secure Elements. These procedures require a mutual authentication based on symmetric cryptographic key. In other words a SE is managed by the entity that controls the Card Manager.

3.2 The Open Mobile API Model

This API [4] defines a generic framework for the access to Secure Elements in a mobile environment. It is based on four main objects.

- The ***SEService*** is the abstract representation of all SEs that are available for applications running in the mobile phone. A service is usually associated to a callback function, invoked by the operating system when the software environment has been successfully created.
- The ***Reader*** is the logical interface with a Secure Element. It is an abstraction from electronics devices which are needed for contact (ISO 7816) and contactless (ISO 14443) smartcards. A *SEService* object gives of list of available Readers.
- The ***Session*** is opened and closed with a Reader. It establishes the logical path with the SE managed by the Reader.
- The ***Channel*** is associated with an application identified by an AID and running in the SE.

There are two kinds of channels: basic and logical. According to the ISO7816 standards SE operating system can manage up to four concurrent applications

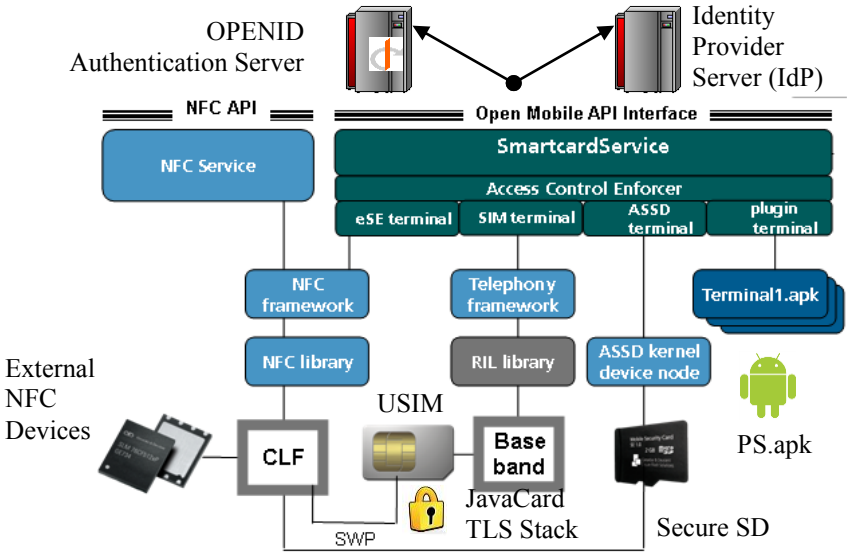


Fig. 4. An illustration of the Open Mobile API environment according to [5], with our identity platform components: USIM TLS stack, Proxy Software -PS-, OPENID and IdP Servers

associated to an index ranging from 0 to 3. This index is included in the two LSB bits of first byte (CLA) of every (APDU) request command. The basic channel is associated to the zero index, other values are refereed as logical channels. In an USIM device the basic channel is allocated to the application that manages authentication (i.e. the AKA algorithm) with the radio network. In other words the Open Mobile API model is application oriented. It manages information exchange with embedded applications (running in the SEs) identified by their AID. This paradigm was previously introduced by the JSR177 API [16] available in Symbian operating systems. Figure 4 illustrates the Open Mobile API services in a mobile environment, which are represented by the green blocks. In this figure CLF means "Contactless Front-end", and is the name used by in ETSI standards [17] for NFC controllers. Three classes of SE are managed. The USIM is accessed thanks to a modified RIL version. A second SE is embedded in a secure SD memory (ASSD standing for Advanced Security SD cards). External contactless smartcards are managed by the NFC (CLF) controller. The idea of trusted application running in mobile is to execute some sensitive procedure in a secure element The Access Control entity (see next section) manages/enforces authorizations for mobile (Android) applications that intend to exchange information with SE.

3.3 Access Control for USIM

In order to protect the USIM from a non-authorized Android application, an access control (AC) mechanism based on the PKCS#15 standard, has been defined in the annex 6 of [3]. According to the ISO 7816 standards repositories (named DF, Dedicated File) and files (named EF, Elementary File) stored in the USIM are identified by two bytes number. The root file (named MF, master file) identifier is 3F00.

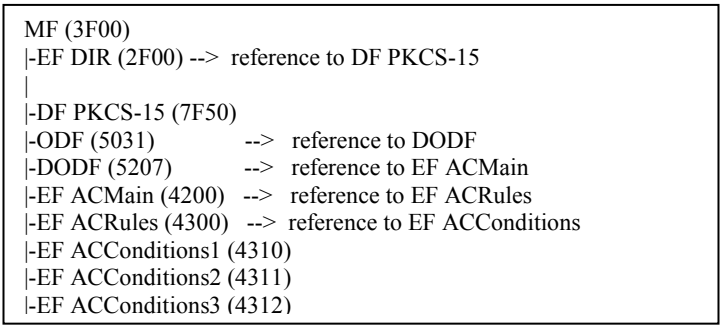


Fig. 5. PKCS#15 files structure for access control

The PKCS#15 repertory (DF-PKCS-15) identifier is listed in the EF-DIR file. Three files defined the access rules (see figure 5):

- The Access Control Main File (EF-ACMF) gives a reference to the Access Control Rules File (EF-ACRF)

- The Access Control Rules File (EF-ACRF) stores a list of Access Control Conditions File (EF-ACCF), each of them being associated to a particular AID.
- Each Access Control Conditions File (EF-ACCF), contains a SHA1 digest of the mobile application whose access to embedded software running in the Secure Element (and identified by its AID) is authorized.

Thanks to this structure the mobile operator defines access rules for mobile application that intend to use services provided by USIM applications.

4 The OPENID Identity Platform

4.1 About OPENID

The OPENID concept was initially developed by Brad Fitzpatrick [21] in 2005, and is based on a specific identity server that *"… is responsible for telling the rest of the world if you're you or not, and digitally signing a receipt saying that you said so, but only if you've told your identity server if you want to*". According to [2] about one million of WEB sites are compatible with this standard in 2012.

OPENID [19] is a Single Sign On (SSO) system made of three entities, the consumer site requiring a user authentication, the authentication server (the OPENID server) performing the user authentication, and an internet user equipped with an IP terminal.

We previously introduced OPENID support for smartcards in [20][24].

The OPENID identifier is an URL that comprises two parts the name of the OPENID server and the user alias. A security association is statically or dynamically established between the consumer and the authentication server, i.e. a secret key is shared between these two entities. It is used for message authentication, performed by an HMAC procedure.

The user gives its identifier on the consumer site, which realizes if necessary a security association with the authentication server. Thereafter a redirection transfers via the user terminal, an OPENID authentication request toward the OPENID server. This last makes the proof of user identity, and returns to the consumer site a list of attributes authenticated by the shared secret.

There are many ways to perform authentication between the user and the OPENID server. The most popular is the simple password mechanism.

In our identity service we use a strong mutual authentication based on a TLS session running in the USIM, in which both client and server are identified by their X509 certificates and associated private keys.

4.2 Components of the OPENID Platform

The Identity platform comprises four elements:

- The USIM equipped with a javacard applet running a TLS stack, and supporting secure facilities for identities downloading.

- A mobile (Android) proxy software (PS), which realizes the logical glue between the browser, the USIM and internet servers. It provides two services, TLS session booting and identity management in the USIM.

- An OPENID server. The mobile is authenticated thanks to TLS sessions running in the USIM. An optional PIN code may enforce a dual form factor procedure.
- An Identity Provider server (IdP). USIM are produced with a device identity. The IdP, whose access are controlled via TLS sessions, generates identities that are afterwards downloaded in the USIM.

4.2.1 The USIM

The USIM stores an application (identified by its AID) and the files needed by the Open Mobile API for the PS authorization.

The TLS services are made available thanks to the EAP-TLS protocol (RFC 2716), introduced in 2000 by the Microsoft Company. These latter transports TLS packets in a datagram like ways, and does not required TCP/IP flavors. The binary encoding rules over ISO7816 APDU are detailed by an IETF draft [26]. This application performs "TLS booting", i.e. it fully handles TLS until the reception of the server finished message for the full mode, and the transmission of the client finished message for the abbreviated (or resume) mode. Afterwards the CipherSuite and the associated KeysBlock values are transferred to the proxy (PS), which handles record layer operations (i.e. encryption/decryption and integrity procedures).

An identity is a set of the following attributes: an alias name; a certification authority (CA) X509 certificate; an X509 certificate; and a RSA private key. The USIM ID is the subject of its current certificate. Each USIM is manufactured with a "root" identity (the device identity or RootID) that cannot be deleted. Nerveless it may store other identities generated by the IdP server, refereed as UserID, and identified by an index greater than zero. They are written in the USIM thanks to ISO7816 commands, and protected by a secure structure named Container. A Container is made of three parts:

- A header, which is the encrypted value of a symmetric AES key, with the RootID public key, according to the PKCS#1 standard.
- A body, which is the encrypted value of all identity attributes, according to an AES-CBC procedure.
- A trailer, which is the signature by a trusted authority (identified by its public key) of the header concatenated to the body, according to the PKCS#1 standard.

Upon downloading, the signature is checked, the AES key is recovered with the USIM RootID private key, and the identity is decrypted and stored. A set of ISO7816 commands realizes the basic operations for identity management, i.e. identity inventory, activation and deletion.

4.2.2 The Android Proxy Software

This Android application realizes the logical glue between the mobile browser, the USIM and internet servers. Its User Interface is illustrated by figure 1. It is possible to enter an optional PIN code and then to start the USIM application with the RootID or the current identity activated in the chip. The PS is a TCP daemon, opened on the loopback address (127.0.0.1), usually with the port 8080. Its interface is made by two kinds of URLs, used for network exchange and USIM commands. The first class of URLs (***ClassI***), such as

http://127.0.0.1:8080/~url=www.server.com/path/file.html

opens an HTTPS session the remote server (i.e. is equivalent to https://www.server.com/path/file.html). The TLS session is booted from the USIM and then transferred back to the PS. As a result the browser opens HTTP session with remote servers that are authenticated via the USIM. The second class of URLs (***ClassII***), such as

http://127.0.0.1:8080/reader/apdu.xml

sends ISO7816 commands to the USIM application. These commands are located in the body of an HTTP POST request. There are encoded as the content of form inputs:

=CMD1&=CMD2&=CMDi

The returned response is encoded according to the XML format.

4.2.3 The Identity Provider (IdP) Server

The IdP server manages the USIM identities. It is in charge of the following operations

- *Subscriber registration/recovering*. Each USIM owns a RootID. The registration process establishes a logical link between the subscriber identity and its RootID. Should the UICC be lost, the subscriber will recover its previously generated IDs, by proving its knowledge of some registration parameters.
- *Identity generation/deletion*. A registered subscriber is always identified / authenticated by its RootID. It may afterwards generate identities for personal use, which are stored in the IdP database.
- *Identity loading/removing*. Identities previously generated may be downloaded in the USIM or removed from the USIM.
- *Identity activation*. Once an identity has been downloaded in the USIM it may be activated, and is afterwards referred as the CurrentID. All further operations dealing with OPENID services work with this identity.

The interactions between the IdP server and the USIM are based on the AJAX [22] technology. AJAX (shorthand for Asynchronous JavaScript and XML) is a set of technologies for executing applications on the browser side, triggered by user's interactions, typically mouse clicks associated with HTML forms, and processed by JavaScript procedures.

AJAX pages are downloaded by the mobile, via ClassI URLs, authenticated by the USIM. These pages include XMLHttpRequest [23] objects that send ClassII HTTP requests to the proxy server. These requests transport ISO7816 commands, executed afterwards by the USIM. The PS returns ISO 7816 responses, embedded in XML pages. These XML contents are parsed and thanks to the DOM (Document Object Model for JavaScript) the initial page is modified, typically with a success status. Thanks to these mechanisms the IdP server downloads container in the USIM, and performs identity management operations (inventory, activation, deletion). All these procedures are available from the mobile WEB browser. Figure 6 illustrates AJAX interaction with the USIM. An ISO7816 command is hidden in an HTML form. When the user clicks on the SetRootID button this form is sent to the proxy software and processed by the USIM. The response encoded in the XML format is afterwards parsed and a modified page is displayed by the browser.

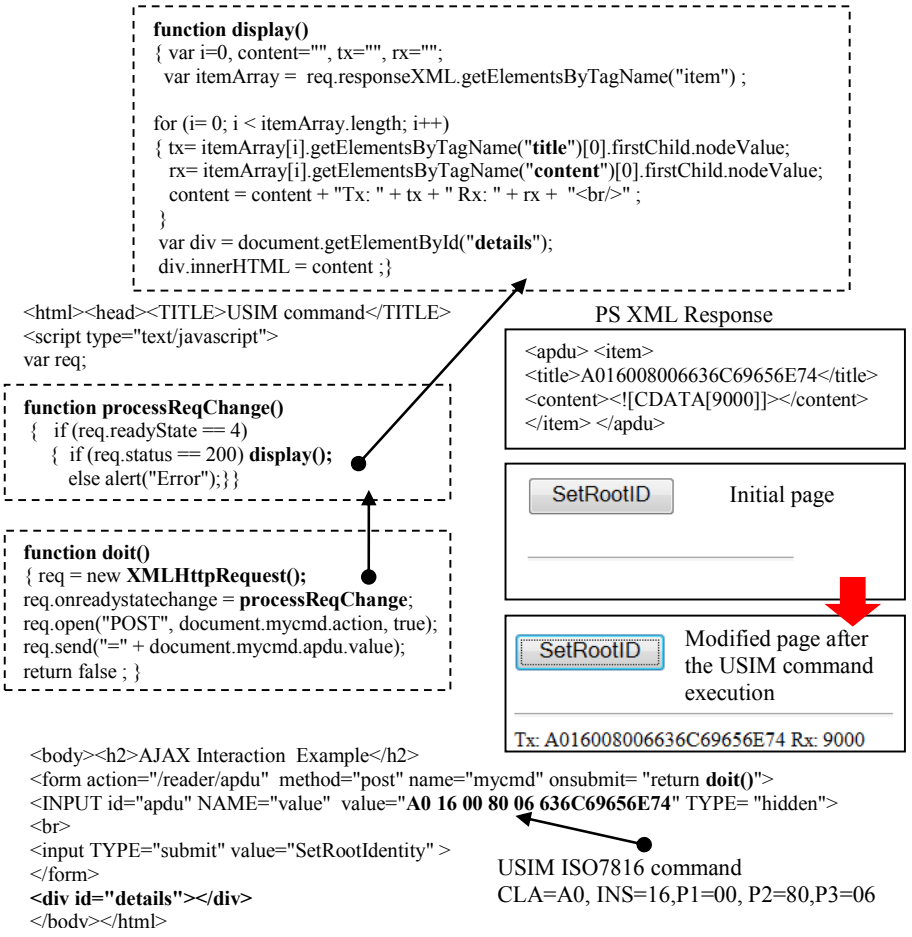


Fig. 6. Example of AJAX interaction with the USIM

4.2.4 The OPENID Server

The OPENID server authenticates the user thanks to its USIM device. It uses ClassI URLs for that purpose. An example of authentication page is shown by figure 1 (part C). This login page is usually downloaded without any security (i.e. with a simple HTTP request). It includes a form typically associated with an action such as:

http://127.0.0.1:8080/~url=www.openid.com/login?Cookie=sid

This URL authenticates the USIM thanks to a TLS session with PKI mutual authentication. For WEB programming issues, it may be necessary to include session id (i.e. the Cookie used by the OPENID server, if any).

5 Conclusion

In this paper we have presented an innovative Identity platform based on OPENID for mobile operator. It has been successfully experimented with Galaxy SII Android phones.

References

1. http://www.gartner.com/it/page.jsp?id=1622614
2. http://trends.builtwith.com/docinfo/OpenID
3. GSM Association, NFC Handset APIs & Requirements Version 2.0 (2011)
4. SIM Alliance, Open Mobile API specification V2.02 (2011)
5. Secure Element Evaluation Kit for the Android platform - the 'SmartCard API', http://code.google.com/p/seek-for-android/
6. http://www.kandroid.org/online-pdk/guide/telephony.html
7. http://source.android.com/source/initializing.html
8. What is android?, http://developer.android.com/guide/basics/what-is-android.html
9. http://www.devicefidelity.com/
10. http://www.tyfone.com/
11. Vedder, K.: "Smart Cards", ETSI Security Workshop (2006), http://www.etsi.org/WebSite/document/Workshop/Security2006/Security2006S1_3_Klaus_Vedder.pdf
12. ISO 7816, Cards Identification - Integrated Circuit Cards with Contacts. The International Organization for Standardization (ISO)
13. Chen, Z.: Java Card™ Technology for Smart Cards: Architecture and Programmer's (The Java Series). Addison-Wesley (2002)
14. NFC Forum, http://www.nfc-forum.org
15. Global Platform, http://www.globalplatform.org/
16. JSR 177, Security and Trust Services API (SATSA) for JavaTM Platform. Micro Edition
17. TS 102 613, Technical Specification Smart Cards; UICC - Contactless Front-end (CLF) Interface; Part 1: Physical and data link layer characteristics (Release 7)
18. RFC 2246, The TLS Protocol Version 1.0, IETF (1999)
19. http://openid.net/
20. Urien, P.: An OpenID Provider based on SSL Smart Cards. In: 7th IEEE Consumer Communications and Networking Conference, CCNC 2010, January 9-12 (2010)
21. http://community.livejournal.com/lj_dev/683939.html
22. Garrett, J.J.: Ajax: A New Approach to Web Applications (February 2005), http://www.adaptivepath.com/ideas/essays/archives/000385.php
23. XMLHttpRequest Level 2, W3C Working Draft 17 (January 2012)
24. Urien, P.: Collaboration of SSL smart cards within the WEB2 landscape. In: CTS 2009 (2009)
25. 3GPP TS 27.007 standard
26. Urien, P., Pujolle, G.: EAP support in smartcards. IETF Draft (2002-2012)

Mobile Training in the Real World for Community Disaster Responders

Natalie Linnell, Ray Bareiss, and Kristoffer Pantic

Carnegie Mellon University, Silicon Valley
Building 23 (MS 23-11), Moffett Field, CA 94035 USA
natlinnell@gmail.com, ray.bareiss@sv.cmu.edu,
me@kristofferpantic.com

Abstract. This paper describes the design and initial evaluation of a mobile application for training Community Emergency Response Teams. Our goal is to model the kind of remediation and performance support provided in high-end eLearning systems, and provide it during hands-on learning in the real world, using mobile phones and sensors embedded in the environment. Thus far we have designed the learning system and tested it with real users, simulating sensor-based activity recognition using an Android-based Wizard of Oz system that we have developed. Our initial user tests found that users were able to use the system to complete tasks, including some that they had never done before. They had little difficulty understanding the interaction mechanism, and overall reacted positively to the system. Though learner reaction was generally positive, these user tests yielded important feedback about ways we can better manage the division between the real world and the digital world.

Keywords: eLearning, mLearning, Ubiquitous Computing.

1 Introduction

The best eLearning systems provide learning-by-doing experiences that allow learners to engage in an authentic task. While completing this task, they receive just-in-time help, advice, and remediation. This methodology is widely regarded as the most effective way to gain "knowledge to be used," as opposed to "facts to be memorized." [1] However, in eLearning, the learner must engage with simulated tasks in the *closed world* of a computer program. This closed world allows the system to easily asses the learner's current state, including their progress in the scenario and any actions they have taken, which in turn allows the system to provide contextually-relevant help, advice, and remediation.

However, computer-based simulations are not appropriate for learning complex tasks such as search and rescue, home repair, or car maintenance; mastering these tasks requires interaction with the *open, real world.* Providing contextualized performance support for such real world tasks requires knowing what the learner is doing in the real world. Until recently, this would have meant requiring explicit input, which can interrupt the flow of the tasks. However, the plummeting price of

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 367–378, 2013.

sensors means we can instrument the environment with sensors to make inferences about the learner's actions. That said, challenges remain in two directions. The first is appropriate learning and interaction design for supporting the learning of complex tasks, using mobile devices to provide context and performance support as the learner interacts with objects in the world. Challenges include determining what will happen in the scenario, on the phone, and what will happen in the real world – and conveying this division clearly to the learner. This has been the focus of our work to date. The second direction, and the primary subject of our future work, is interpretation of rich sensor data to derive context, that is inferring actions from position, motion, services accessed, and other sensor data. Since our work in activity recognition from sensor data is still in progress, we have developed a Wizard of Oz system for Android [2]. This system allows us to simulate activity recognition by manually controlling the content a user sees on their phone, from a remote view on the experimenter's laptop.

The main contribution of this paper is an exploration of the interaction design space at the intersection of the real world and the digital world that occurs when the learner is acting in the real world but receiving context and remediation on a mobile device. When the project is completed, another contribution of this work will be bringing together work in activity recognition from sensor data and indoor locationing with our work in design for this unique but interesting interaction mechanism.

We are conducting this research in the context of developing training for Community Emergency Response Teams (CERT). CERT members are volunteers who mobilize in the event of a disaster to gather data about damage in their neighborhood, and to perform triage. Tasks CERT members must perform include assessing building damage, performing medical triage of victims, turning off utilities, and light search and rescue. These are tasks in which recognition of actions and inference of learner intentions is critical to successfully providing contextualized help, advice, and remediation and in which appropriate learning and interaction design are essential to ensure that such support aids rather than distracts the learner.

2 Background on CERT

Community Emergency Response Teams (CERTs) are composed of civilian volunteers with minimal training who self-organize by neighborhood. The purpose of CERT is to activate in the wake of a disaster, such as an earthquake, and to gather information about damage and injuries to pass along to professional responders, and to carry out basic search and rescue and first aid services when professional responders are unavailable or overburdened.

In the event of a disaster, all CERT members assemble at their neighborhood's pre-defined gathering point to establish a command post. Since utilities are often affected in an emergency, CERTs train for all communication to happen via radio. Small 2-3 person teams are sent out to canvass the neighborhood and collect information using the Damage Assessment Form (DAF), communicating back to the command post via FRS radios (i.e., walkie-talkies) when they encounter an urgent situation.

The core of CERT activity is filling in the DAF (Fig. 1). This form is used to collect information about damage to houses and buildings in the neighborhood, as well as about injured or trapped individuals at those locations. A "master" copy of the DAF is maintained at the neighborhood command post.

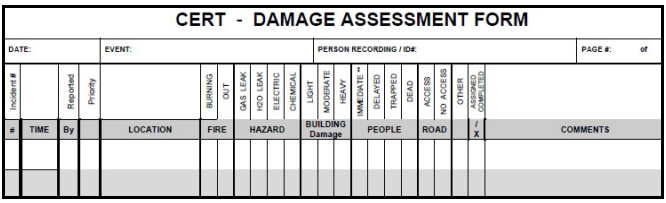


Fig. 1. The paper DAF (10 more empty lines not shown)

We realized that it would be easier for us to provide advice and remediate reporting mistakes if we used a mobile phone version of the DAF and integrated it into the system; as it turns out, there are a number of other benefits to a mobile DAF (see Sec IV). For this reason we developed a mobile version of the DAF (Fig. 2a), and structured the training around it. This form also includes built-in performance support in the form of "Help" screens (Fig. 2b).

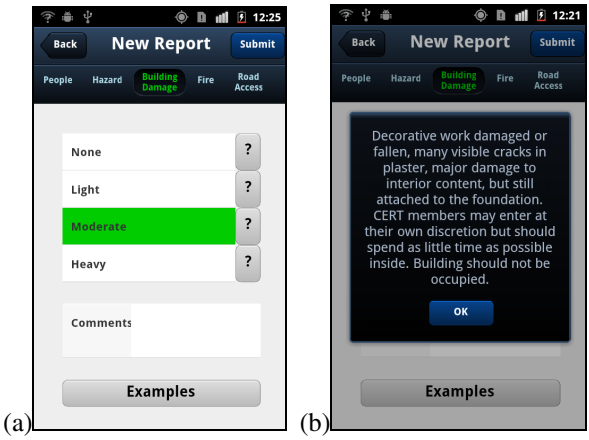


Fig. 2. The mobile Damage Assessment Form

After 20 hours of training from the city, any further training or organization is left entirely up to the volunteers themselves. This means that the ongoing training and procedures put in place can vary from one CERT group to another. Typical training would include having teams canvass an area, and report incidents that they make up on the spot to the command post, with little or no practice in hands-on skills such as victim evaluation or evacuation. Benefits that we foresee for our system are the possibility of helping to standardize training and practices across different CERT groups, and to enable more extensive training without requiring additional professional resources. However, the fact that CERT organizers are volunteers means that our system must be easy and inexpensive to deploy – there will be no technical support person to ease adoption on-site.

3 The System

A number of different components of the system interact to provide the full training experience

- *Conveying the scenario to the learner*. We use video and sometimes low-fidelity augmented reality in the form of modified photos that simulate damage.
- *Mobile version of DAF*. CERT practice centers on the DAF so it is helpful to put the form on the phone.
- *Performance support materials*. These provide contextualized help and advice when the learner is unsure how to proceed.
- *Activity recognition from sensor data.* This will enable us to detect learner actions.
- *Remediation materials*. These materials are presented when the learner makes a mistake.

Some of these have been fully realized in the current version of the system and some are still in progress. In particular, the activity recognition component is currently the primary focus of our future work.

Since we decided to target design of the system before robust activity recognition was in place, we have developed a Wizard of Oz tool for Android [2]. Wizard of Oz is a technique via which a remote experimenter can push content to a user's screen to simulate unimplemented functionality. In our case, the experimenter watches the learner, and when they see the learner take an action that would trigger an activity recognition event, they manually send the appropriate content to the learner's phone. As our system progressed beyond the prototype stage, we continued to use the tool's underlying communication system to force transitions on the learner's phone from an experimenter's laptop, thus simulating more sophisticated activity recognition than is currently implemented.

An example will clarify the components' functionality. This example describes how a pair of learners, participating in training, might interact with our system in its current state, in the context of the scenario from the user test discussed in Section 5.

3.1 Example Usage Scenario

Sally and Bill are our learners. They will be working together using one phone for receiving scenario information and reporting. Their experience begins with a video on the phone that orients them to the application and the scenario. The video presents a narrator voice-over of images of the interface as the interaction mechanism is explained, including introducing a chime sound that is played when content on the phone requires attention. After the video, they are given the option to play the first scenario video, or take some time to get ready before starting the training.

Since they feel like they understand the technology, they play the first scenario video. The actor in the video introduces himself as the neighborhood incident commander, and instructs the learners to carry out a damage assessment of their assigned area, gathering appropriate information about each house. The incident commander informs the learners that they will be working in a small team. A new person, also an actor, comes onscreen and introduces himself as Tom, the third member of their team, who will assist them with their damage assessment.

Tom reminds them that before they start the damage assessment, the team should be sure they have all the necessary equipment. The video ends, and Sally and Bill are presented with a checklist of the equipment that they should have with them when doing a damage assessment. After Bill and Sally have completed the checklist, Tom is shown saying "Let's get started."

The team then knocks on the door of a house that has been instrumented for training. The experimenter pushes a video. The learners hear the chime sound introduced in the orientation video, and the video starts. Tom is shown knocking on the door, loudly identifying himself as CERT. From inside, we hear a victim shout "Is someone there? Help! I'm trapped!" Tom explains that before entering, the team must assess the building damage. The victim sounds flustered, and asks them to hurry. This video highlights one of the most difficult tensions for CERT members: that between personal safety and helping victims. If the building damage is too heavy CERT members should not enter the building – their first priority is personal safety.

Bill and Sally begin to walk around the house. When they reach a pre-defined spot, the experimenter pushes a picture of the house that has been altered to include a large puddle of water in the front yard. They record a water hazard in the DAF. As they round the corner of the house, the experimenter pushes a picture of the house that has been altered to show the chimney pulling away from the house. Sally opens the DAF to record the level of building damage, but is unsure if this should be classified as moderate or heavy damage. She touches the "?" button next to the entries for moderate and heavy on the form, and reads the descriptions (Fig. 2). After reading the descriptions, she and Bill think the damage is probably moderate, but still aren't quite sure. They push a button at the bottom that says "Examples" and realize that what they are seeing is much closer to the picture of moderate damage than the heavy damage example. They determine that the damage is moderate, and Sally records this in the DAF. As they continue walking along the side of the house, they near a practice gas valve that is not attached to the house. The experimenter pushes a transition, and a video is shown. Tom says "I smell gas. Will one of you turn off the gas valve?" Sally uses her wrench to turn off the gas valve and Bill records the gas leak on the DAF.

Having determined that the building damage is moderate, and so it is all right to enter the building for a short time, Bill and Sally decide to enter the house. When they reach the front door, the experimenter pushes a video of Tom identifying himself again, and reminding the learners to mark the building before entering. Bill and Sally can't remember exactly what should be included in the building marking, so they start to enter without marking the building. The experimenter pushes a diagram showing what information should be included in a FEMA standard marking, and after studying it Sally and Bill mark the building using a piece of chalk, then enter the house. Upon entry, they see a dummy trapped under a bookcase, and the experimenter pushes a video. The video shows a victim trapped under a bookcase that has fallen on his leg (Figure 2). Tom asks the victim some questions to rule out the possibility of a back or neck injury. He then asks the learners to use cribbing (a technique where one person lifts the bookcase using a prybar, and another person puts wood under the bookcase to hold it up) to elevate the bookcase, then to use a blanket drag to get the victim out of the damaged building. After the video is over, a screen asks if they'd like more information about cribbing or blanket drags. Sally and Bill select "Cribbing" and are shown a diagram of a cribbing operation. They notice some cribbing materials

available in the room and begin to raise the bookcase, stopping occasionally to consult the diagram. Once it is raised enough, they pull the dummy out from under the bookcase. They go back to the phone and choose to see more information about blanket drags. Step-by-step instructions are shown, alongside a diagram. They roll the dummy onto the blanket, then drag the dummy outside.

Fig. 3. Video showing a trapped victim (the screen image is simulated)

Once outside, the experimenter pushes a video of Tom doing a shock assessment of the victim and asking the learners to record the information gathered in the DAF. Once the learners have entered the appropriate information into the DAF, the experimenter pushes a video of Tom telling the victim that help will be there soon. The scenario ends with a video of the incident commander congratulating the learners on a job well done.

3.2 Activity Recognition

We are just beginning to integrate activity recognition into our system, so in the user test described Section 5 we relied on Wizard of Oz transitions to simulate all activity recognition. However, we did use the test as opportunity to gather information about how learners interacted with objects in the scenario. We gathered data on the levels of acceleration achieved when: the front door opened, the dummy was moved, and the prybar was used. As a first step in activity recognition, we are simply detecting when objects that we have instrumented with sensors move, and the data we gathered during the user test will allow us to set intelligent movement thresholds; with this data moving to this first level of activity recognition should be achieved very quickly. More sophisticated activity recognition, focusing not only on whether an action was taken, but whether it was done correctly, is the main focus of our future work. Eventually we would like to replace all Wizard of Oz transitions in the above scenario with activity recognition. For example, we would like to determine whether the dummy has moved, and if it was moved correctly, preventing harm to the victim. We are also exploring using infrastructureless indoor locationing techniques to get specific location data [3].

For sensor packs, we have decided at this stage to use Android phones, which are inexpensive and widely available. This means that we attach a Android phone to any item that we wish to track. Our main motivation in beginning with very simple activity recognition and in choosing this sensor platform was deployability. We were

initially concerned that size and cost would be limiting factors in deploying Android phones as sensor packs, however existing wireless sensor packs (e.g. Sunspots) were not significantly smaller, and were in fact more expensive than used last-generation Android phones (which are available for about $35 on eBay). In addition, when we think about deploying the system, a CERT is likely to be able to acquire old Android phones on their own, making it possible for any CERT group to deploy simple sensor-enhanced training scenarios. Of course, Android phones have limitations as sensor packs – for example, it would be unwieldy to attempt to use one on a small tool such as a hammer – so as part of future work we are planning to work with small custom sensor packs [4]. However, we prefer to do as much as possible using the Android phones, because of the ease of programming and deploying them.

4 System Design

We have previously written about the learning theory of doing eLearning in the real world [5]. Building on this work, we engaged in an intensive design cycle for the mobile DAF, the performance support for the form, and the training scenario we used in our proof-of concept.

4.1 Mobile Version of the Damage Assessment Form

As mentioned earlier, we realized that it would be much easier for us to remediate reporting mistakes if we migrated to a mobile phone based version of the DAF and integrated it into the system. However, this would not make sense if it was not likely to be adopted outside the training. As it turns out, there are multiple benefits of a mobile version of the DAF that make CERT members and city officials that we have talked to enthusiastic about a mobile version of the DAF for its own sake. It simplifies the reporting process, since even if cell towers are down, information can be sent using NFC, Bluetooth, etc. In addition, the current design of the paper DAF can be confusing, as it was designed to optimize analysis rather than reporting.

We also implemented performance support in the form of "Help" screens associated with the DAF. For instance, if a learner chooses the "Help" button on the "People" tab, they are taken to a form that walks them through the steps of assessing a victim, and the results are saved in the top-level DAF. We have also included step-by-step instructions for actions such as building marking and turning off a gas valve.

For design of the DAF, we brainstormed several interaction frameworks before choosing the tabbed interface we implemented. We then did a small user study with three CERT volunteers, where we received valuable feedback that led to significant changes, including a different tab layout and a free text comment field on each tab.

We undertook a similar design process for the performance support materials, culminating in another small user test with three CERT volunteers. The main outcome was the importance of keeping the learner oriented within the larger DAF while using the performance support materials.

4.2 Conveying Scenario Context

We iterated extensively on the scenario, getting several rounds of feedback from the director of the CERT program in our city. Since the biggest challenge in CERT is

assessing a situation and acting accordingly, it is vital that our training system be situated in a realistic scenario. We convey the scenario using video on the mobile phone and occasionally low-fidelity augmented reality (photos of the location where the training takes place that have been altered to reflect damage), and then the learner is asked to act in the real world in response to the scenario snippet they have just seen. For example, in our scenario a CERT member is shown a video of a victim trapped under a bookcase, and then be asked to remove a dummy from underneath a bookcase in the real world. While removing the dummy would be a useful training exercise without the scenario, the video allows us to provide context that situates the mechanics of moving the bookcase within the decisions that must precede extracting a real victim; Do they have spinal injuries? What are the risks associated with staying in the damaged building? etc.

In designing the scenario, it was our goal to place as much of the action in the real world as possible. To bridge the gap between the actions the learner must take in the real world and the actions in the video, we included Tom, the "virtual" member of the team. Tom is a CERT member, like our learners, but he undertakes the actions that require interaction with the aspects of the scenario that are different from the real world; e.g. Tom smells the gas leak and asks our learners to turn off the gas valve.

5 User Test – Design and Results

We performed a preliminary user test of our system, using the scenario described in Section 3. We conducted 6 user tests. We conducted three user tests with active CERT members; two user tests were conducted with pairs and one was a single user (all were intended to be pairs but one user cancelled). We also conducted three single-user tests with non-CERT members. All 8 of our users worked in the technology industry and were very technologically savvy.

We chose to conduct these two kinds of tests (pairs of CERT members and single non-CERT members) to understand the range of learners who could realistically be trained using our system. CERT members usually work in groups of 2-3 during neighborhood training exercises, so testing with pairs of active CERT members is representative of the interaction we might expect if our system was used in neighborhood training. But we were also interested in judging the feasibility of using our system for non-CERT trained people who are interested in becoming involved, but do not have time to complete the 20-hour training course. People fitting this description routinely join neighborhood training exercises - could our system be adapted to provide a useful first-time learning experience to these people? If so, what changes or additions would be necessary?

5.1 Discussion of Tests with CERT Members

In our first user test, the learners did not fully appreciate the difference between the form and the scenario content. This led to confusion when they were expected to navigate through the form to fill in information; when the video ended, they were returned to whatever part of the form they had been looking at before the video, which was not necessarily relevant to the video they had just seen. However, they assumed that whatever screen was present when the video ended was the appropriate screen to

fill in in the form. In all subsequent tests, we took an additional introductory moment to emphasize the separation between these two components of the system, and none of the other learners experienced significant difficulty around this issue.

However, some other learners did express a desire to be able to interact with the form and the scenario materials simultaneously, e.g. fill out the form while watching a video. Both of these points highlight an interesting fact; we had initially conceived of the learner interacting with two different worlds: the real world and the digital world. We found that they are in fact interacting with three worlds: the real world, the digital scenario, and the digital form which reflects the scenario, but is the same form that would be used in a real disaster. This points to the possible utility of the pair of trainees using two phones during the training, one to relay scenario materials, and one for filling in the form. Alternatively, we could make greater visual distinction between the scenario and the form.

We chose to have pairs of learners share a single phone, both for ease of coordination, and to provide a single focal point. The learners had no issues seeing the content in this configuration, and we speculate that it actually improved collaboration. Another interesting issue is how directive the scenario should be. That is, should the scenario lead the learner step-by-step through the correct actions, or should intervention be minimized to only those times when the real world does not match the scenario world? We chose the latter, and found that learner reaction to this choice was mixed. One pair tended to be very passive, waiting for the next piece of content to be pushed to them to tell them what to do, and both this group and the singleton learner expressed the desire for more explicit instructions. The other pair became very immersed in the damage assessment, and expressed a desire that more of the action take place in the real world. This points to the possibility of providing a minimal amount of scenario content that is *pushed* to the learner, but making additional content available for the learner to access if desired (for instance, we could include a "What do I do next" button that tells the learner explicitly what the next activity should be.) This would also be helpful for new learners.

When asked, all of the learners said that they preferred having scenario information presented in video form rather than in text form (as is normally done during a neighborhood exercise). However, there were some problems encountered in working with video on a mobile phone. One was sound. Most CERT activities take place outside, and it was often difficult for learners to understand the audio. In addition, we found that even when they didn't have a hard time understanding the audio, they did not catch all of the pertinent information in the video, and rarely did they re-watch the video to get it. This points to the possibility of using text to highlight the important points of the video, either during the video or after it is complete.

5.2 Discussion of Tests with Non-CERT Members

Tests with non-CERT learners were conducted slightly differently. We prepared a number of very directive prompts (e.g. "Let's start by knocking on the door") which could be sent if the learner got stuck, and pushed support materials proactively; e.g. when the learner was expected to turn off the gas valve we immediately pushed instructions instead of waiting for the learner to request them. We wanted to see whether these small accommodations would be sufficient for non-CERT learners to complete the scenario and if not, what additional accommodations were necessary.

Somewhat surprisingly, two of the three subjects completed most of the scenario with relatively little difficulty. Through heavy reliance on the support materials, these learners were mostly able to correctly assess damage, turn off the gas valve, record hazards, and evacuate the "victim" (Fig. 4). The main difference was the heavy reliance on support materials. One learner did say that his attention was focused on the phone to an extent that he thought detracted from his ability to focus on the real world.

The third learner struggled with the division between the scenario and the real world. This learner was highly dependent on directions, at times requiring verbal instruction from the experimenter, and missed some opportunities to interact with the real world. In spite of this, the learner was very positive about the system. One thing that might help with learners who become overly focused on the phone would be to have less scenario content up front, requiring the learner to interact with the real world in a highly structured way early in the experience, to establish the primacy of interaction with the real world. Since only one of the three learners had major issues with the interaction mechanism, it seems likely that if we asked new learners to complete the scenario in pairs, this issue could be overcome. In fact, this solution would likely address most of the above issues with testing with new learners, especially if new learners were placed in a group with an experienced CERT member. Further testing will be required to validate this hypothesis.

An issue identified by all of these learners was that of terminology. These learners were unfamiliar with terms like "damage assessment" and "cribbing." Since these are terms that are often used in CERT, we find it preferable to address this issue by defining these terms as they arise, rather than by replacing them.

Some other misunderstandings arose during these user tests. One learner created a new report for every hazard, not realizing that a single report should correspond to a single location. Two other learners failed to explore all of the outside area. However, these kinds of mistakes seem simple to address either with additional background materials, or by having learners work in groups containing at least one experienced CERT member. Overall, we found this exploratory test very encouraging as to the possibility of using a slightly enhanced version of the system as an initial teaching tool, especially if learners work in pairs or groups.

Fig. 4. The learner uses a blanket drag on the dummy

6 Related Work

As noted earlier, the main focus of our future work is integrating sensor data regarding context and activity recognition into our system. Researchers in the ubiquitous learning community have discussed the utility of a range of sensors in

establishing context (e.g., [6,7,8]), but implemented learning applications tend to use only location, and sometimes proximity to tagged objects, as proxies for rich sensor-determined context. The majority of these previously developed solutions focus on learning tasks that tend to be somewhat unfocused (e.g., learning about plants in a botanic garden [8] or paintings in a gallery [9]) and are similar in spirit to a number of GPS-based tour guide projects (e.g., [10]). A few projects do attempt to teach more structured tasks (e.g., performing single-crystal X-ray diffraction [11] or assembling a computer [12]); these too rely on location, proximity to tagged objects, and learner input to determine what guidance to provide.

In parallel, researchers in the ubiquitous *computing* community have been researching the recognition of actions from sensors (e.g., [13,14,15]). Their approaches use a mix of techniques drawn from statistical machine learning and natural language processing to create recognition models and to apply them to segmented sensor data. Various inference techniques are used to integrate data from different sensors for higher-level recognition (e.g., [16]).

7 Future Work and Conclusions

We have decided to take a staged approach to activity recognition from sensor data. This allows us to begin with very simple activity recognition that we believe an individual CERT could realistically deploy and then enhance the technology to find a balance between deployability and functionality. For this reason, we will begin with simply detecting when objects move; with the threshold data we gathered in our user test, this step will require little additional programming to integrate with our system. Beyond this, we have been talking to colleagues about adapting their existing activity recognition systems for our purposes [15]. We are also exploring the possibility of using infrastructureless indoor locationing to obtain accurate location data [3].

In addition, we recognize the limitations of using Android phones as sensors. In particular, instrumenting small tools such as wrenches would greatly expand our ability to understand learner actions but requires significantly smaller and more robust sensor packages – which means we must use custom-made solutions. We have been discussing collaboration with colleagues working on such a sensor [4].

This paper describes the design and initial evaluation of a system for training Community Emergency Response Teams. Our user tests found that learners reacted positively to the system, and the fact that even experienced CERT members struggled with the tasks presented in the hands-on training and benefitted from performance support suggests that this training fills an unserved need. The most challenging aspect of designing the learning experience has been managing the boundary between the real world and the scenario. Our user tests suggest that for the most part this was done successfully, but also elucidated possibilities for allowing learners to move this boundary to suit their tastes and experience levels. In particular, we found that learner preferences for the amount of direction in the training varied widely, and were not purely a function of CERT experience, so it would be useful to include ways to allow the learner to adjust the level of direction. When testing with non-CERT trained learners, we were surprised by the level of performance they were able to achieve with relatively simple performance support and remediation. This suggests that our system could find wider use than we had initially conceived, possibly providing initial training to new CERT members.

Acknowledgments. We thank the members of Carnegie Mellon's Disaster Management Initiative, especially Lynn Brown, Martin Griss, Roger Miller, and the members of Mountain View CERT.

References

1. Bransford, J., Brown, A., Cocking, R. (eds.): How People Learn: Brain, Mind, Experience, and School. National Academy Press, Washington, D.C. (2000)
2. Linnell, N., Bareiss, R., Pantic, K.: A Wizard of Oz Tool for Android. Presented as a poster at Mobile HCI (2012)
3. Nguyen, T.L., Zhang, Y.: Probabilistic Infrastructureless Positioning in the Pocket. In: Zhang, J.Y., Wilkiewicz, J., Nahapetian, A. (eds.) MobiCASE 2011. LNICST, vol. 95, pp. 311–330. Springer, Heidelberg (2012)
4. Mokaya, F., Kuo, C., Zhang, P.: MARS: A Muscle Activity Recognition System Using Inertial Sensors. In: ACM/IEEE Conference on Information Processing in Sensor Networks (ISPN) (2012)
5. Linnell, N., Bareiss, R., Griss, M.: Contextualized Mobile Support for Learning By Doing in the Real World. Ubiquitous Learning Journal (2012)
6. Wang, Y.: Context Awareness and Adaptation in Mobile Learning. In: Proceedings of the 2nd IEEE International Workshop on Wireless and Mobile Technologies in Education (WMTE 2004), pp. 154–158 (2004)
7. Hwang, G., Tsai, C., Yang, S.: Criteria, Strategies and Research Issues of Context-Aware Ubiquitous Learning. Educational Technology & Society 11(2), 81–91 (2008)
8. Beale, R., Lonsdale, P.: Mobile Context Aware Systems: The Intelligence to Support Tasks and Effectively Utilise Resources. In: Brewster, S., Dunlop, M.D. (eds.) Mobile HCI 2004. LNCS, vol. 3160, pp. 240–251. Springer, Heidelberg (2004)
9. Lonsdale, P., Baber, C., Sharples, M., Byrne, W., Arvanitis, T., Brundell, P., Beale, R.: Context awareness for MOBIlearn: Creating an engaging learning experience in an art museum. In: Attewell, J., Savill-Smith, C. (eds.) Mobile Learning Anytime Everywhere: A Book of Papers from MLEARN 2004, pp. 115–118 (2004)
10. Davies, N., Cheverst, K., Mitchell, K., Efrat, A.: Using and determining location in a context-sensitive tour guide. IEEE Computer, 35–41 (August 2001)
11. Hwang, G., Yang, T., Tsai, C., Yang, S.: A context-aware ubiquitous learning environment for conducting complex science experiments. Computers & Education 53(2), 402–413 (2009)
12. El-Bishouty, M., Ogata, H., Yano, Y.: Personalized knowledge awareness pap in computer supported ubiquitous learning. In: Sixth International Conference on Advanced Learning Technologies, pp. 817–821 (July 2006)
13. Kwapisz, J., Weiss, G., Moore, S.: Activity recognition using cell phone accelerometers. ACM SIGKDD 12(2), 42–43 (2010)
14. Aipperspach, R., Cohen, E., Canny, J.: Modeling human behavior from simple sensors in the home. In: Proceedings of IEEE Conference on Pervasive Computing, pp. 337–348 (April 2006)
15. Chennuru, S., Chen, P.-W., Zhu, J., Zhang, J.Y.: Mobile Lifelogger – Recording, Indexing, and Understanding a Mobile User's Life. In: Gris, M., Yang, G. (eds.) MobiCASE 2010. LNICST, vol. 76, pp. 263–281. Springer, Heidelberg (2012)
16. Patterson, D.J., Liao, L., Fox, D., Kautz, H.: Inferring High-Level Behavior from Low-Level Sensors. In: Dey, A.K., Schmidt, A., McCarthy, J.F. (eds.) UbiComp 2003. LNCS, vol. 2864, pp. 73–89. Springer, Heidelberg (2003)

Shop Social: The Adventures of a Barcode Scanning Application in the Wild

Jonathan Engelsma[1], Ferris Jumah[1], Alejandro Montoya[1], Joseph Roth[1], Venu Vasudevan[2], and Greg Zavitz[1]

[1] School of Computing and Information Systems
Grand Valley State University
Allendale, MI 49401 USA
[2] Motorola Mobility
Libertyville, IL 60048
jonathan.engelsma@gvsu.edu,
{ferris.jumah,alejomontoya,roth.joseph.e,gzavitz}@gmail.com,
venu.vasudevan@motorola.com

Abstract. Mobile retail is a space rich with plausible hypotheses but sparse on longitudinal datasets that give us a corpus of user behavior to validate or disprove theories around the use of digital devices in a physical store. Popular price comparison apps such as ShopSavvy have shown that a smartphone in the aisle is a reality that brick-and-mortar retailers have to contend with. A nuanced, data-driven understanding of a smartphone powered shopper might enable store-based retailers to leverage the smartphone rather than fear it as something that leads to sales erosion. To this end, we built and deployed a novel, mobile retail app that blends mobile, media and social capabilities. In this paper, we describe the user needs and design axioms behind the app, and the data that we've collected over the course of its use by about 5,500 users over the period of a year.

Keywords: mobile, retail, barcode, video.

1 Introduction

The rapid proliferation of smartphones and their associated application ecosystems is reshaping the way consumers shop for products and interact with the retailers who sell them. Smartphone horsepower, 3G+ bandwidth and the availability of high performance barcode scanner apps have made smartphones a powerful comparison shopping tool in the brick-and-mortar aisle. So much so that the retail industry has begun to take steps against showrooming [1], the phenomenon of shoppers using brick-and-mortar stores as a showroom for getting a better look at the product, while ultimately purchasing the product online.

While the retail industry has treated showrooming as a threat, this project began with the hypothesis that retailers might be able to use technology to turn that threat into opportunity for complex, priced goods. In particular the hypothesis was that

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 379–390, 2013.

beyond price sensitivity, showrooming demonstrates an unmet need amongst consumers to be better educated about products. The superior performance of Best Buy (until recently) demonstrated that within the confines of a single category (brick-and-mortar in this case) retailers that were able to educate, enlighten and de-risk a product to their prospective customer, were more effective in sell through and ultimately overall retail sales.

We wanted to explore whether the combination of mobile, social and rich media on a smartphone could approximate a well-trained sales force in this regard, at a fraction of the cost. The goal was to detect a user's interest in a product (i.e. barcode scan), and to trigger a rich media conversation around the product experience with authentic user generated content. Additionally, social capabilities integrated into the app would enable the user to extend this visual conversation to his social group, a group that social retail research has shown to be highly influential in driving purchase decisions.

Our goal in creating Shop Social was to use an 'in the wild' application to gain concrete insight on the confluence of mobile, social and rich media via in-market learning. In particular, we wanted to understand the relative proportion of usage of these facets amongst various population segments of shoppers, and the potential of app design alternatives to drive adoption of specific usage patterns.

Recently, barcode scanning apps have gone well beyond the basic price comparison concept. [2,3,4,5,6]. However, most of the widely adopted barcode scanning apps focus primarily on price comparison. In the typical use case, the user scans a product's universal product code (UPC), and receives back a list of retailers (both online and near the user's current location) and the price for which each retailer offered the product. We deliberately chose to avoid the price comparison feature, as we wanted to offer users functionality that didn't already exist in other barcoding applications. We also wanted to approach this from the mindset of a brick and mortar retailer, and hence assumed that a price comparison feature would likely not be a priority. We decided early on that the key distinguishing feature of the application would be to focus on delivering the user just-in-time relevant product video based on a barcode scan.

We also noticed that to the best of our knowledge, none of the barcode scanning applications at that time integrated well with social media platforms. Intuitively, it seemed that relevant just-in-time video content along with the ability to interact around products and product content within a user's social graph could be just as compelling as price comparison, and particular helpful for a consumer making product purchase decisions.

In the remainder of this paper we describe the application itself, and the rationale for the various design and technical decisions made in creating it. We provide a brief analysis of over a years worth of application and analytics data collected by the application in the wild, and conclude with a discussion of the lessons learned.

2 Shop Social: An Experimental Barcode Scanning App

Shop Social is an experimental barcode scanning mobile application that locates product information and relevant product video content based on barcodes the user scans. Shop Social leverages the user's social graph along with product-related media

(product descriptions, reviews, videos, photos, etc.) to help a shopper understand a particular product's potential for them, or for whomever they might be shopping. In an attempt to further encourage engagement, Shop Social also incorporates simple badging game mechanics.

In addition to the native application running on user handsets is a set of non-trivial network services that feed content to the client applications and also serve as a communication conduit to enable users to interact together within the application. In what follows we describe the end-to-end architecture of the application including both the network services deployed in Google App Engine, and the native client applications that were developed for iPhone and Android.

2.1 The Shop Social Backend

We had a number of goals/constraints that influenced the approach we took in architecting the Shop Social experience. These included:

- *Minimum Operational Cost*: Ideally, we wanted to able to deploy and support up to 10K users without incurring hosting fees or at least keeping the costs as low as possible.
- *Scalability*: While we are a university lab without a lot of marketing muscle to promote the app, we wanted our implementation to be capable of auto-scaling up to large numbers of users without having to re-architecture our backend should the opportunity arise.
- *Persist Client Data in the Cloud*: We wanted to keep the client applications as simple as possible, caching data locally only when necessary for performance reasons. A further benefit of having the data in the cloud is that it gave us the visibility we needed as researchers to understand how users were using the application.
- *Integrate with an Existing Social Network Destination for User Authentication and Access to Social Graph*: It made sense to piggyback our application experience on top of an existing social network destination. Not only did this simplify the services we had to build and support in the backend, but it also made it easier for users to automatically discover which of their friends were already using the application. It also allowed users to share application content with friends who were not already using the app, and thereby extending the reach of the application. Facebook was the obvious social platform to integrate with.
- *Utilize Existing Product Data and Product-Related Media*. Not being a retailer ourselves, we had no product database of our own. Fortunately, there are large existing databases of product information searchable by UPC via web service interfaces that can be used to retrieve product metadata. Over the course of the project we integrated with several different product databases, and are currently using Google's Search API for Shopping. YouTube's public search APIs coupled with some simple heuristics was used to find relevant video content. We used Flickr APIs to search for relevant product photos.
- *Adopt a "platform" Approach*: Our goal was to make it as easy as possible to swap out architectural components moving forward. e.g. keep all Facebook-related

integration in the backend within a single adapter, so in the future a different social destination could be used with minimum impact on the code. All sources of product data were integrated via a generic adapter component, so we could readily switch to another database in the future with minimal impact on our code base. Paying attention to this early on turned out to be very important as we switched product data sources several times to-date.

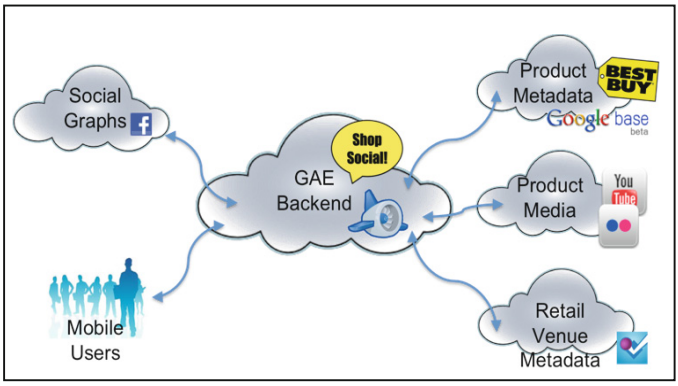


Fig. 1. The end-to-end Shop Social Architecture

Figure 1 above provides a high level view of the end-to-end Shop Social architecture. It consists of the following components:

Mobile Users: End users install the native Shop Social application via the appropriate application store (Apple iTunes App Store and the Google Play Store) on their mobile phone, and then proceed to scan product barcodes and interact with product content and other users via their Facebook friends. Users of the Shop Social are allowed to interact in two different modes: anonymous and authenticated. In anonymous mode, the basic app features are available such as scanning products, viewing relevant videos, etc. If the user authenticates they can also keep track of their favorite products, share app content, view their social dashboard, and earn badges. Sharing extends the reach of the app beyond the users who have installed it on their phone. For example, Facebook users who may not be using the Shop Social application on their phone, might interact with the application when they encounter content generated on-line (e.g. a user recommends a product or comments on a video posted via the app on another user's Facebook wall, via the Shop Social mobile app.) Our backend system also tracked these sorts of interactions on Facebook by having all links in Shop Social postings redirect through our server.

GAE Backend: The Shop Social backend services were implemented on Google's AppEngine (GAE) platform. The backend component coordinates and persists all end user interactions in the application. This includes mobile users running the Shop Social client application as well as online users who happen upon content generated by the application in their Facebook activity stream. The backend also is responsible for efficiently generating a user's social "dashboards" on demand, and locating

relevant product media (videos/photos) via a set of heuristics that operate on the product/media metadata as well as the historic user interaction data persisted over time.

Social Graphs: In terms of social network destinations, the current backend integrates with the Facebook platform. The backend integration is maintained as a generic "adaptor" to facilitate future integrations with other social platforms.

Product Metadata: When products are referenced, via UPC scans, clicked on from social dashboards or from scan history lists information on that particular product needs to be retrieved from the network. Currently, the backend adopts a two-step process. First, the backend performs a lookup via Best Buy's Remix API the first time it sees a new UPC code. If no data is found there, the backend then "punts" the request to the Google Search API for Shopping. Remix provides better and more consistent metadata for many popular consumer electronics products, as well as user reviews.

Product Media: Currently the backend utilizes the public web APIs of YouTube and Flickr to locate videos and photos relevant to a particular product. This process will be described in more detail below. Once again, the backend integration with these services has been carefully factored out into a set of adapters that can easily be exchanged with alternative product media sources moving forward. The client integration (e.g. when photos/videos are actually viewed on the client) is also decoupled (e.g. operates on URLs) and would not be difficult to direct to a different media source.

Retail Venue Data: The application attempts to encourage in-store interaction around products by offering various badges for in-store interaction. In addition, the application locates retail venues near the user's current location and encourages them to check-in. The latest revision of the client utilizes the FourSquare Venue API and provides a wide range of nearby retail establishments to the end user. Backend integration is kept to an absolute minimum, in that it only needs to tag interactions with venue ids when they occur within a store, as well as a track of how many times each user visits a particular venue in order to award badges.

Google Analytics Instance: Though not shown in Figure 1, both of the client apps are carefully instrumented with Google Analytics to generate detailed analytics as user interact on the clients. The data collected by Google Analytics as well as the user activity stream persisted in our GAE backend will be analyzed in the next section.

2.1.1 Locating Relevant Video Content

Relevant videos for a given product are discovered via heuristics that take both product/video metadata into account as well as prior user video interactions and the requesting user's social graph. The social component is based on the intuition that videos discovered in the context of a given product, are likely to be more relevant to you if your friends interacted with that same video in the past.

The video search begins by locating previous video interaction for the same barcode, and scores every video v previously associated with a UPC code or that turns up when YouTube is searched with the product name. The score s_v for each video v in this initial set of videos is computed as follows:

$$s_v = YTW_{name}(v,pname) + \sum_{k \in A_{v,upc}} SW(k_{age}, k_{type}) * FW(uid, k_{uid})$$

where YTW_{name} is a weight function which returns a positive non zero value if v turns up in a YouTube search on product name (pname), or zero otherwise. $A_{v,upc}$ is defined as the set of all the past activities that involved video v in the context of the product upc, and SW is an initial weighting based on the age and type of the activity. Possible activity types in $A_{v,upc}$ can be view video, share video, or rate video, and each of these is weighted differently. FW is a weight function that will be a positive non-zero value if the activity k is tagged with a user k_{uid} that happens to be in the social graph of the current user uid, and returns zero if the activity was not tagged by a friend. The idea behind SW is to take into account the nature of the past interactions, and also to gradually reduce their importance as they age. The FW function is the component that incorporates social relevance into the scoring.

Once this initial set of videos is located and scored, it is sorted and the N highest videos are selected. If there are < N videos total at this point, then YouTube is searched with increasingly broadening terms, initially by a more sanitized version of the product name (e.g. one that has punctuation, etc. stripped out), followed by a search by product category, followed by a search by product manufacturer. Each video that turns up in these broader searches is scored by a constant weight that decreases as the search broadens. If a video turns up from the set of previously scored videos described, the scoring is accumulative. After each addition search, the current list of candidate videos is sorted, and if there are < N total, the search continues.

This approach almost inevitably returns videos. The situation in which no videos are discovered has been discussed. One possible action in this situation is to simply return a list of obviously irrelevant videos (e.g. Charlie Chaplin black and whites) with a message "Sorry, nobody seems to have made a video of this product, but we thought you'd might enjoy these instead." Since in practice this rarely occurs, at the moment if this were to happen the user simply gets no videos.

Photo content was added to the application as an afterthought and at the moment we simply do a brute force search on Flickr using the product name. In practice we've noticed the photos are often irrelevant and this is confirmed by the analytics data capture, as very few users have interacted with the photos the application turns up.

2.1.2 Efficiently Generating the Social Dashboard

GAE's persistence layer is optimized to handle requests for a single entry. It is not convenient to aggregate data or request a set of random entries. To display the social tab in Shop Social, the application needs to query the database based on a user's list of Facebook friends that is dynamically fetched from a Facebook API. Internally, App Engine converts the SELECT IN query on the array of user ids into a separate query for each user id which is not going to be very efficient in our context. According to Facebook, the average user has 130 friends [7]. Running 130 queries is not an optimal use of resources, and some individuals have thousands of friends. In most cases, when bootstrapping a new application like this, it's likely that most of the user's friends are not yet using the app, so a significant amount of work would be done to return a result of a small number of "hits".

To reduce the number of queries against the datastore, the user ids of authenticated Shop Social users are stored via a Bloom filter [8]. A bloom filter runs a series of hashing functions to set bits in a bit string. The basic idea is to tolerate very infrequent false positives for enormous efficiency gains. The Bloom filter is persisted in GAE's memcache for fast access and backed up in the datastore for integrity. Before querying the datastore with the array of user ids, they are passed individually to the bloom filter to identify which ids are in the datastore. The test in the bloom filter is quicker than a query against the datastore and it also does not count towards GAE quotas. Since the bloom filter uses a hashing function, there is a small probability for a collision that will produce a false positive. False positives have negligible impact since the filtered list of user ids will be queried against the datastore to obtain the actual friend data and an occasional invalid id will still result in the correct data and only cost an extraneous access to the data store.

Before using the bloom filter, users with over 500 friends could experience significant lag when loading the social tab. Occasionally, the request would exceed App Engines CPU time limit and fail to complete. Even with smaller numbers of friends, the backend would quickly consume its daily free quota and we'd end up with a bill from Google. With the implementation of the Bloom filter to pre-screen the friend data queries, dashboard loading is very efficient and typically independent of the number of Facebook friends a user has, as most users have a relatively small number of friends actually using the application.

2.2 The Shop Social Mobile Application

The Shop Social mobile application was implemented for both the iPhone and Android phones. The iPhone version of the application was made available to the general public for free in the Apple iTune App Store. The Android version was made available for free via the Google Play Store, as well as the Amazon App Store.

The application supports a variety of features including, product lookup by scanning or keying a product UPC. The UPC is forwarded to the GAE backend which returns the product description, product reviews, and a list of relevant videos and photos. Recently scanned products are retained temporarily in the scan history, and the user also has the ability to favorite the product adding it to their "My Stuff" list.

The user's social dashboard consists of a list of their Facebook friends, a gallery of badges that each friend has earned to-date, and a list of their favorite products. All of the functionality available to users when they scan a product directly is also available to users when they encounter products on their social dashboard. In addition to watching product videos, users are able to share the videos via Facebook, and/or rate videos with a thumbs-up or thumbs-down as shown in Figure 2. The fact that a user watches a video and/or rates it is ultimately taken into account in future video relevancy scoring as described in the previous section.

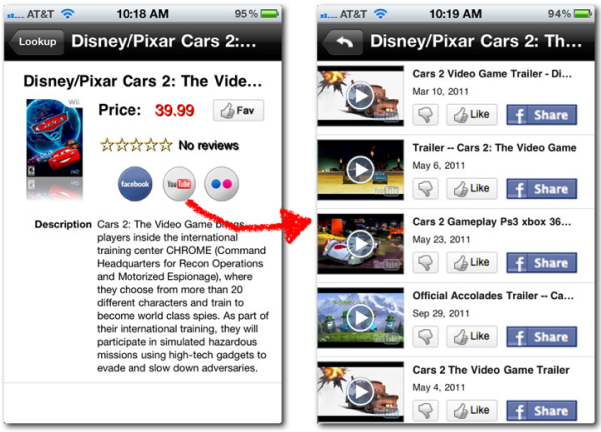


Fig. 2. A barcode scan of the Cars 2 Video Game brings up product data and a list of relevant YouTube videos

We were also interested in learning whether or not people would interact with product information via the application while they visited brick-and-mortar retail establishments. In order to accomplish this the application uses its current location to search the FourSquare backend for nearby retail stores. Since FourSquare has many non-retail venues, the lists of nearby venues obtain from FourSquare was filtered by venue type, and only retail businesses were displayed. Users could then "check-in" to the venue, much like they do in the FourSquare app, and all subsequent product activities (as long as the user stayed in the general vicinity) would get tagged with that venue id. In order to incentivize check-ins, the user who checks in to the venue the most becomes its "Best Badger", which is analogous to the FourSquare concept of "mayor". We also introduced a variety of additional badges to try to encourage users to engage with the application.

3 Results

This section describes what has been learned from the usage data that the application collected via Google Analytics as well as the data captured on the Shop Social backend.

3.1 User Participation

The total number of unique mobile users from 15 October 2010 – 23 November 2011 was 5,499. Of those, 3,804 of them were using Android devices, and 1,695 were using iPhones. Of the overall user base, 527 of them actually authenticated with their Facebook credentials, which allowed them to interact with friends using the app and share application content with friends on Facebook whether or not they were using Shop Social. This had an impact on how the users interacted with the app, which will be discussed in the next section. The chart in Figure 3 below shows the total number of unique users using the app each day during that range. The tall sharp peaks early on in the Android plot correspond with updates in the respective application stores.

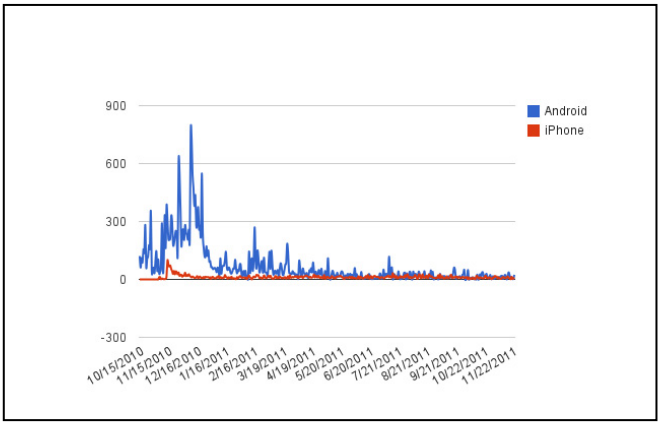


Fig. 3. Number of participants overtime using Android and iPhone smartphones

3.2 User Interactions with the Application

A user interaction with Shop Social is referred to an "activity". In the current implementation, the following activity types are defined and tracked:

- scan: a barcode is scanned.
- video_view: a video associated with a product is viewed by the user.
- share: an item (product, video, badge, photo, etc.) is shared by a user via Facebook.
- checkin: a user checks into a retail venue.

Figure 4 contrasts the frequency these various activities occurred. Barcode scanning is by far the most popular activity, followed by video viewing. These two activities are available to users independent of whether or not they authenticate with Facebook. Sharing and checkin activities were only possible once users authenticated.

In terms of the breakdown of activities over the two population segments of the user base (authenticated vs. non-authenticated), the former had a much higher activity level than the latter. The authenticated users who represent around 10% of the overall user base generated approximately 43% of all activities. Hence, authenticated users of the application are more engaged than non-authenticated users. While we've learned from this and other applications we've deployed in the wild that it is very important to allow users to use the application in meaningful ways without authenticating, this data suggests to us that it is important to incentivize user's to authenticate with Facebook or some other popular social destination. Intuitively, once a user authenticates with Facebook in Shop Social they can begin to see what product content their friends are interacting with, which presumably is more engaging to most users than product information void of any social data.

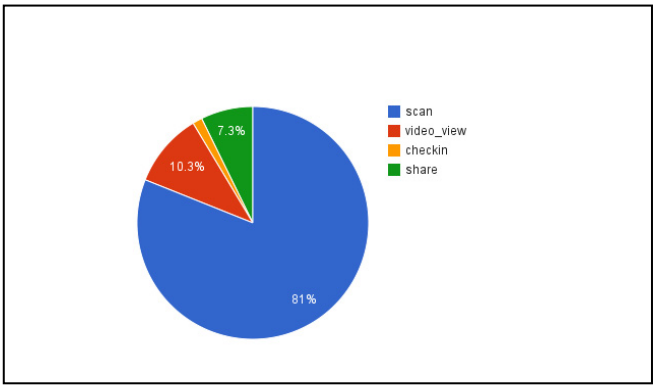


Fig. 4. Breakdown of the types of activies users performed

3.3 Social Sharing

While scanning barcodes was by far the most common activity in which users engaged, users did actively share product content they encountered while using the application. Approximately 80% of the shared items were products, followed by videos (17%). The remaining 3% of the shares were product photos and badges earned by users.

We also broke down the products that appeared in the aggregate activity stream over the past year into specific product categories to determine if a particular type of product was interacted with more than another. Figure 5 shows the various product categories that the application activities were involved in and shows in particular the percentage of actions overall in a particular product category as well as the percentage of overall share-related activities in that category. Looking at the data in this way we can see that while products in the food, book, and health & beauty categories generated the most interest in general, users are more inclined to share information in the consumer electronics and video games categories. One possible explanation for the dominance of the book and food related activities is the fact that the UPC codes on these products typically persist overtime on the food container or book cover. When users load the application up for the first time, the nearest barcode is probably on the nearest book or food container.

Another interesting set of data the backend captured was the response to the shared data. When users shared product and video information on their Facebook activity streams from the application, any click-through events by any Facebook users, regardless of whether or not they were Shop Social users, were redirected through the Shop Social backend. The data showed that video content (essentially product relevant YouTube videos) though shared less, than products themselves, on average received more click-through response per share than the product shares (essentially a product thumbnail image with link to a webpage with more info on the product).

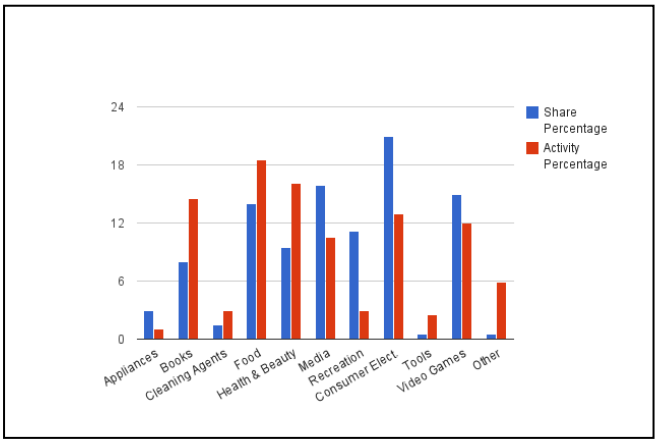


Fig. 5. Aggregate activity vs. share activity by product category

4 Conclusion

Shop Social has been a fascinating learning experience from both a technical perspective as well as an experience design perspective. While we embarked on this journey as an attempt at gaining more practical insight into how a brick-and-mortar retailer might utilize mobile technology, the project ended up taking a life of its own. Along the way we learned a number of lessons that we feel transcend the retail problem space we were looking at, and apply to many different contexts in which one is building mobile applications and deploying them in the wild. Some of the specific lessons learned include:

1. Engaging end users in a mobile experience is very difficult. The vast majority of the users who have downloaded the app seem to bounce, never to be seen again.
2. Android is the more interesting platform from a mobile applications research perspective. We believe part of the success we had with Android was due in part to the speed at which we could iterate on the experience design and immediately deploy in the Google Play Store. The bump in usage every time a revision became available was very consistent. Not only can problems be addressed swiftly, but ideas can be tested and tinkered with. This cannot be done nearly as effectively on iPhone given the approximate 8 days it takes for revisions to make it through the Apple curating process.
3. In terms of social sharing, a few interesting shared videos are worth a thousand product shares. This point was made earlier, and is worth noting once again. While it could be argued we haven't yet collected enough data, it does seem intuitive and is supported by what we've seen in the data so far. We will keep monitoring this moving forward.

4. Automating product relevant video search on YouTube is reasonably feasible, even without a significant amount of participation. We think the current heuristic works quite well, without a lot of usage data to optimize with. It should get even better with more usage.
5. While attracting iPhone users is a more difficult proposition than attracting Android users, the data collected in this experiment indicates that engaging iPhone users beyond the initial app impression is easier than engaging Android users. This may be due to the more consistent and appealing user interface of the iPhone platform.
6. While its important for apps to offer utility to anonymous end users, authenticated users are more engaged users. In our study, around 10% of the users generated 43% of the activity. This seems to suggest a couple of very important guidelines to app developers. a) Make sure there is a meaningful return on investment for authenticating. b) Make authentication as pain free as possible.

References

1. Eason, G., Noble, B., Sneddon, I.: 'Showrooming': People Shoppin in Stores, Then Researching By Cell Phone, Says Pew Survey (January 31, 2012), http://abcnews.go.com/Technology/pew-internet-showrooming-half-cell-phone-users-research/story?id=15480115#.T90OJitYui2
2. Dorman, K., Yahyanejad, M., Nahapetian, A., Suh, M.-k., Sarrafzadeh, M., McCarthy, W., Kaiser, W.: Nutrition Monitor: A Food Purchase and Consumption Monitoring Mobile System. In: Phan, T., Montanari, R., Zerfos, P. (eds.) MobiCASE 2009. LNICST, vol. 35, pp. 1–11. Springer, Heidelberg (2010)
3. Adelmann, R.: Mobile Phone Based Interaction with Everyday Products - On the Go. In: Proc. of 2007 International Conference on Next Generation Mobile Applications, Services and Technologies, pp. 63–69 (2007)
4. Bulusu, N., Chou, C., Kanhere, S., Dong, Y., Sehgal, S., Sullivan, D., Blazeski, L.: Participatory Sensing in Commerce: Using Mobile Camera Phones to Track Market Price Dispersion. In: Proceedings of UrbanSense 2008 (2008)
5. Gao, J., Kulkarni, V., Ranavat, H., Chang, L., Hsing, M.: A 2D Barcode-Based Mobile Payment System. In: Proc. 3rd International Conference on Multimedia and Ubiquitous Engineering, pp. 320–329 (2009)
6. Deng, L., Cox, L.: LiveCompare: Grocery Bargain Hunting Through Participatory Sensing. In: Proc. 10th Workshop on Mobile Computing Systems and Applications (2009)
7. Burbary, K.: Facebook Demographics Revisited (March 7, 2011), http://www.facebook.com/note.php?note_id=197149076992338
8. Bloom, B.: Space/Time Trade-Offs In Hash Coding With Allowable Errors. Communications of the ACM 13(7), 422–426 (1970)

The *webinos* Architecture: A Developer's Point of View

Paolo Vergori[1], Christos Ntanos[2], Marco Gavelli[1], and Dimitris Askounis[2]

[1] Istituto Superiore Mario Boella, MultiLayer Wireless solutions (MAIN)
Turin, Italy
{vergori,gavelli}@ismb.it
[2] National Technical University of Athens,
Electrical and Computer Engineering Department
Athens, Greece
{cntanos,askous}@epu.ntua.gr

Abstract. This work describes the architecture proposed by the *webinos* EU project, which aims at developing software components for the future Internet, in the form of Web Runtime Extensions. It discusses the *webinos* architecture from a developer's point of view, presenting an overview of its main advantages, such as context-awareness capabilities and distributed APIs in an intrinsic secure environment. It also shows how these features can in practice prove beneficial to the development of ubiquitous and secure web applications based on standard technologies like HTML, CSS and JavaScript.

Keywords: webinos, mobile applications, middleware, developers, framework, architecture, context, context awareness, distributed apis.

1 Introduction

The world of web development is rapidly changing. Requirements for developers are more strict and involve scenarios that were not taken into account in the past. These changes are mainly due to the steep increase of the range of available devices, their features, their capabilities and the need for persistent network connectivity. Furthermore, in this new world of distributed information, validating the same user across diverse devices and Operating Systems can be challenging with respect to security and privacy.

These factors are increasingly forcing developers into creating cross-platform applications, with cross-service capabilities that are able to share information among devices. Such features, though, introduce new security requirements that need to be addressed. In the end, the aim is to provide appropriate tools for the development of web applications that augment the user experience, while at the same time, relying on sand-boxed environments, like browsers, and on Operating Systems' integrated security.

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 391–399, 2013.

In this paper[1] we aim to point out how the *webinos* consortium addresses those needs and simplifies development tasks, by introducing an innovative, open-source middleware, to the development process. The *webinos* framework enhances application security, regardless of the Operating System on which the framework is running, without requiring any additional effort from the developer. In the following sections, we present an overview of other related solutions and the current state-of-the-art for middleware web based applications and we follow it with a brief breakdown of the *webinos* architecture. Next, we break down the main strengths of the *webinos* approach, and point out some of the key findings from the research on context-awareness, as it is implemented in *webinos*.

2 Background

As a cutting-edge middleware framework, *webinos* is closely related to the research on the development of middleware for distributed applications, a topic that is well document in literature. In general, middleware components aim to be generic across applications and industries, run on multiple platforms, be distributed, and support standard interfaces and protocols.[1, p. 5]

It is widely suggested that *'conventional binary programs will be limited to system software, whereas the vast majority of end user software will be developed using web technologies'*[13]. Due to the already well established ubiquity of web browsers, the *webinos* framework was steered early towards serving its functionality through web content. Best practices from W3C [15] refer to a Web Application as a *'Web page (XHTML or a variant thereof + CSS) or collection of Web pages delivered over HTTP, which use server-side or client-side processing (e.g. JavaScript) to provide an 'application-like' experience within a Web browser. Web applications are distinct from simple Web content, in that they include locally executable elements of interactivity and persistent state.'*

Despite the fact that numerous technologies that can be delivered through HTML, such as JavaScript, CSS and DOM models, have enabled a more dynamic environment for Web applications, web pages, in general, are still providing an insufficient experience to nomad users. This is due to the fact that ubiquity of information and functionality has not been taken seriously into consideration yet, nor is data exchange aware of the context in which it is performed.

In order to enhance the already established specifications with an outlook towards the future of computing, the *webinos* consortium quickly underlined the importance of identifying user requirements that would cover present, but more importantly, future needs. The result was a series of original use-cases and scenarios, where the framework would play a key role in enabling developers overcome hurdles associated with the diversity of the modern and future device ecosystems.

[1] The work reported in this paper was granted by the *webinos* project co-funded by European Union Seventh Framework Programme.

Web Applications vs Widgets

The distinction between web applications and widgets has been thoroughly covered by a previous investigation from webinos partners [8], but it is equally relevant from the application developer's point of view. The W3C identifies[16, p. 3] a widget as a pre-packaged, purpose-built web application. Moreover, in this specification[16] it is explained that web applications and widgets *'differ from HTML5-style web applications in the sense that for web widgets, the installation formats and practices have been predefined specifically to resemble those of traditional (native) applications. For instance, each W3C web widget is packaged into a separate WGT (ZIP) file that is installed explicitly in the same fashion as binary applications. This is different from HTML5 applications that run in a web browser without explicit installation.'*

It is clear that web applications are not prepackaged for installation, they do not offer an automated installation procedure, nor a maintenance system, and therefore they are static. Widgets can be more appealing for adoption by web developers, especially in the case of distributed computing applications like the ones that the *webinos* framework serves, because they provide a dedicated distribution package that delivers contents to users, adding an intrinsic value to the contents themselves.

3 Overview and Overall Architecture

The overall architecture of the framework that the *webinos* consortium is delivering is briefly depicted in the following section.

3.1 What Is *webinos*?

Webinos is an EU funded project aiming to deliver a platform for web applications across mobile, PC, home media and in-car devices. The webinos project has over twenty partners from across Europe spanning academic institutions, industry research firms, software firms, handset manufacturers and automotive manufacturers. *webinos* is a 'Service Platform' project under the EU FP7 ICT Programme. The main features of *webinos* are:

- *webinos* bases on the achievements of the web community and extends an open source web runtime environment
- *webinos* offers a common set of APIs to allow easy access to cross-user cross-service cross-device functionality in an open and secure manner
- *webinos* aims at easy programming of applications by offering a single virtual device that can consist of all devices owned by a user
- *webinos* creates open specifications and open source reference implementations that show the feasibility of the specifications and simplify their adaptation by the industry [17].

3.2 The Concept of the Personal Zone

The Personal Zone concept stands on top of the *webinos* architecture. The *Personal Zone Area (PZA)* is delimited by the perimeter defined by the ownership of devices. Each *webinos-enabled* device that belongs to the same user is considered to be inside this perimeter.

Each *PZA* is defined by a single point of synchronization, which all devices must authenticate against. This point is called a *Personal Zone Hub (PZH)* and its aim is to provide attestation, authentication and to act as an enforcement point for incoming requests. The actual implementation relies on a cloud positioning of the personal hub, assuming it is hosted securely, even if it's on a third party server. A set of available actions is provided to the *PZH* user, through a dedicated web UI to which the user logs-in via a custom OpenID [11] authentication mechanism. The available actions through this interface include adding a device to the *Personal Zone Area*, revoking certificates for every authenticated device in that area and checking their authentication status. The certificate revocation action is irrevocable and another certificate must be issued again from the device itself, before being issued to the *PZH* again.

Figure 1 depicts the architecture of *webinos* and the components with which an application can interact.

Every *webinos-enabled* device is running an instance of a *Personal Zone Proxy (PZP)*. The connection with the *Personal Zone Hub* is done through a secure TLS channel that mutually authenticates both sides, but in case of out-of-band communication, it can peer itself with an already authenticated *PZP* that belongs to the same *PZA*, maintaining security policies, due to the duplicated nature of the *Policy Enforcement Point* and *Policy Decision Point* in both *PZH* and *PZP* [7]. Its main role is to export device APIs as services in a completely secure manner. *Webinos Service Discovery* allows these services to be discovered and consequently be invoked using Remote Process Calls (RPCs) based on the JSON-RPC 2.0 Specification [4]. The *Personal Zone Proxy* connects to the *Webinos RunTime (WRT)* with a customized websocket implementation, called *secure channel*. The *WRT* is the place where all the developed applications belong and run. It has a *Widget Manager* for the installation, deletion and updating of applications that are packaged as widgets.

Making use of this architecture and the resulting development environment, the software engineer has the opportunity to build applications with the significant advantage of having all the information and the *webinos-enabled* device features in a distributed and secure overlay network. Moreover, all personal zones have context-awareness features enabled and as it is explained in section 4, developers can create advanced ubiquitous applications that can rely on the *webinos* framework.

3.3 Breakthroughs Introduced by *webinos*

Several similar frameworks have already been deployed in the past few years. They all have tried to address the cross-device concept on a distributed environment. Most, if not all, of these frameworks are missing one or more crucial

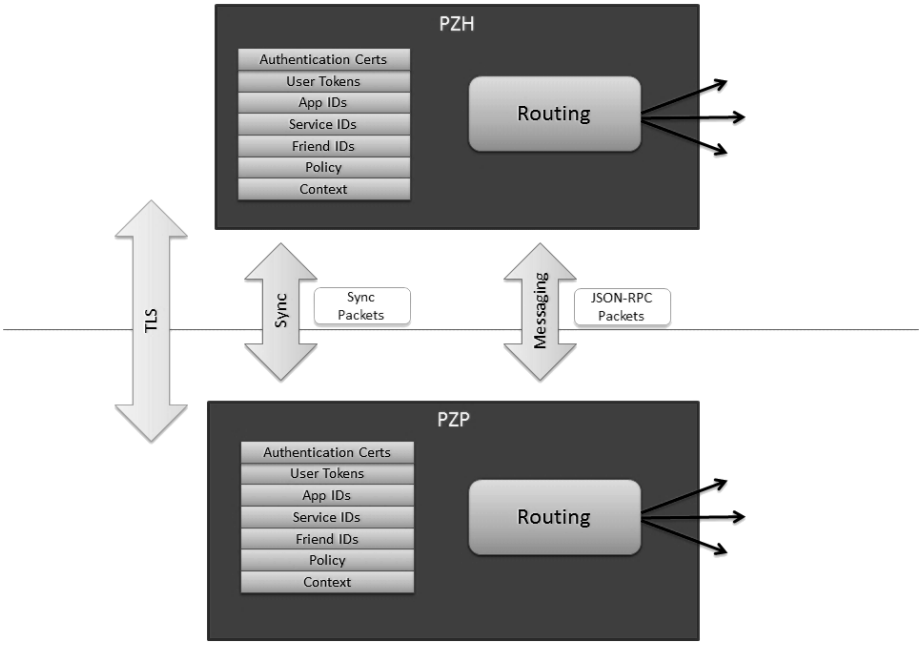


Fig. 1. The *webinos* architecture

functions that *webinos* aims to either ameliorate or introduce. PhoneGap [10] provides a precompiled framework that serves a platform dependent API subset, and runs on several Operating Systems. This approach lacks a pure multi-device, platform-independent development concept. Moreover, the compatible OSes are limited to the mobile device pool. Bondi is a web-based Operating System just for mobile devices. It creates a secure environment, in which it allows access to mobile functionality from a widget renderer. In July 2010, OMTP and the BONDI initiative were subsumed into the Wholesale Applications Community (WAC) [9]. The Wholesale Applications Community (WAC) is an organization aiming to create a unified and open platform to help mobile software developers easily write applications that can run on a variety of devices, Operating Systems and networks [14]. Both Bondi and WAC are closely related to the *webinos* Project. Nevertheless, *webinos* is more ambitious, due to its aim to overcome the mobile barrier by aiming to reach a wide variety of devices, such as set-top boxes, PCs, cars and, of course, tablets and smartphones as well.

The main goal of the *webinos* framework is to take computing to its next logical step, that of ubiquity. In order to do so, knowing the state of the device and the user at any given time, and making decisions based on that context (section 4) is crucial.

Context awareness and true cross-platform and cross-device applications cannot be achieved without the developer having access to a distributed API environment. With the use of a framework that provides such access, together with the context awareness capabilities of *webinos*, 'everyware' [3] applications are a step closer to reality.

Applications operating within a *webinos* PZA can share the state of the same application on any other device belonging to the same user, by seamlessly sharing context and application data across the personal zone. Moreover, the use of Remote Process Calls [4] from the *Webinos Run Time* to the exposed APIs of other devices within the PZA, will allow the developers to limit the functionality of their applications almost only my their own imagination.

4 Context and Distributed Information from Developers POV

The *webinos* platform aims to provide appropriate tools for software engineers, in order to enable and facilitate the development of cross-platform context-aware applications. *Context-aware computing* is a term introduced by *Bill Schilit* in 1994 [12] and was later popularized by the advancements of mobile computing, device, software and telecommunication interoperability. In computing, context can be described by its three important aspects: *where you are, whom you are with, and what resources are nearby.* Because of this, context is divided into three parts, *computing context* (e.g. available processors, devices accessible for user input and display, nearby resources) , *user context* (e.g. users location, collection of nearby people, user profiles and social situation) and *physical context* (e.g. lighting, temperature, noise and humidity level, traffic conditions).

4.1 Context-Aware Applications

A context-aware application either makes use of a single piece, a combination, or a time series of context related data, for example, the current state of the compass on a mobile phone, the current state of the ambient light sensor of a laptop computer or the geographic location and the closest Wi-Fi networks, in order to make an *automatic contextual reconfiguration* (e.g. increase the brightness of the monitor, enable Wi-Fi connectivity) or enable a *proximate selection* (e.g. highlight Points Of Interest geographically located near the user or print a document on the closest printer). In order to take advantage of such functionality, an application developer has to have access to means of *acquiring* contextual data, *storing* them, *filtering* them, *combining* them and *performing commands* based on the resulting information. For ubiquitous, distributed and context-aware computing applications, the aim is to provide appropriate *middleware* that can perform Remote Process Calls (RPCs), while at the same time introducing an *abstraction layer* that will facilitate the development process, by hiding the heterogeneity of the networking environment, supporting advanced coordination models among distributed entities and making as transparent as possible the distribution of computation [5].

4.2 Context-Awareness in *webinos*

The *webinos* project aims to provide a cross-platform level of abstraction for procedural calls, but at the same time, incorporate an additional data abstraction layer for use in third party context-aware and context-sharing applications that are *webinos-enabled*. The main data construct relating to contextual information in *webinos* is the *Context Object*. Inspired by the definition of a *meme* [2], a *Context Object* is a unit for carrying data that *uniquely defines a piece of contextual information*. For example, whereas a call to a GPS sensor will return a number of outputs (latitude, longitude, heading, speed, accuracy and altitude accuracy), one relevant Context Object that can be called *MyLocation* will contain only the most relevant data that can define the unit, by excluding some and/or adding others, in this case ending up with latitude, longitude, accuracy, altitude accuracy and time. Context data collection can be performed in three ways. First, there is an automatic mechanism that, with the permission of the user, can *intercept RPCs* made by *webinos-enabled* applications to the various *webinos* APIs. Second, Context Objects can be registered for *periodic background data collection* when the PZP is running and third, they can be d*efined and stored independently by any application*. Apart from the Context Objects already defined for the *webinos* APIs in the *webinos API Context Vocabulary*, a list of structures and rules for the automatic contextualization of intercepted RPC messages, an application developer can make use of the *webinos Application Context Vocabulary* to define custom rules and structures for storing application-specific Context Objects, or ones that are derived by any process or combination of preexisting or new contextual data. The database where these objects are stored securely is located at the user's PZH. The *Context DB* contains data from across the devices and applications in a PZA and each database is unique for that PZA. Querying the Context DB is achieved through a simple to use dedicated query builder that allows the treatment of the Context DB as an Object-Oriented Database, focusing in its main construct, the Context Object. The developer can perform queries directly to the Context API, with the prospect of acquiring any type of Context Objects, created by any API, any application and any device across the user's PZA.

4.3 Contextual Information and Privacy in *webinos*

It is very clear that users are not very good at understanding the future value of keeping personal information private [6] and are often quick to share the ownership of such information without evaluating the impact of its possible uses. With this in mind, the *webinos* platform ensures that the ownership of contextual information stays with the user, while access rights to applications to store, extract or query context data can be given by the user to the application and not the application developer. This allows the developer to build applications that can utilize Context Objects that are stored by the *webinos* platform or other applications in the same PZA. In order to further secure the privacy of personal data, all transactions with the Context DB are monitored by the *webinos* policy

manager and specific access rights to read/write, to and from the Context DB are provided *per application* and *per type* and *per source* of Context Objects.

5 Conclusions

It is becoming increasingly apparent that in order to open the gates of the Future Internet era, applications will be required to be mobile, secure, platform independent and context-aware. Practical and unrestricted ubiquitous computing is gradually leaving the theoretical sphere and is becoming a tangible requirement from end-users. Application developers that need to address these requirements do not have the resources to provide cross-platform implementations of their software or even different builds per Operating System and/or device. Handling cross-device communications, security, privacy, hardware changes, Operating System updates and advanced features, such as context-awareness is next to impossible in most cases. The innovative, open-source and holistic solution that is introduced by *webinos* is a significant step towards real ubiquity of computing, where the software developer will not have to worry about any of these issues, but rather focus on what is really important in developing unique and useful applications: his own ingenuity.

Acknowledgments. The research included in this work has been co-founded by the European Union Seventh Framework Programme (FP7-ICT-2009-5 Objective 1.2) [17].

The researchers that with their work have made this paper possible and deserve a special mention are: Riccardo Scopigno (scopigno@ismb.it), Michele Morello (morello@ismb.it), Nadir Raimondo (raimondo@ismb.it), Enrico Baccaglini (bac-caglini@ismb.it), Andreas Botsikas (abot@epu.ntua.gr) and the rest of the *webinos* team.

References

1. Bernstein, P.A.: Middleware: a model for distributed system services. Commun. ACM 39(2), 86–98 (1996)
2. Dawkins, R.: The Selfish Gene. Oxford paperbacks. Oxford University Press (2006)
3. Greenfield, A.: Everyware: The Dawning Age of Ubiquitous Computing. Peachpit Press, Berkeley (2006)
4. JSON-RPC Working Group. Json-rpc 2.0 specification, `http://www.jsonrpc.org/specification`
5. Issarny, V., Caporuscio, M., Georgantas, N.: A perspective on the future of middleware-based software engineering. In: 2007 Future of Software Engineering, FOSE 2007, pp. 244–258. IEEE Computer Society, Washington, DC (2007)
6. Jedrzejczyk, L., Price, B.A., Bandara, A.K., Nuseibeh, B.: On the impact of real-time feedback on users' behaviour in mobile location-sharing applications. In: Proceedings of the Sixth Symposium on Usable Privacy and Security, SOUPS 2010, pp. 14:1–14:12. ACM, New York (2010)

7. Lyle, J., Monteleone, S., Faily, S., Patti, D., Ricciato, F.: Scross-platform access control for mobile web applications. In: IEEE International Symposium on Policies for Distributed Systems & Networks (2012)
8. Lyle, J., Faily, S., Fléchais, I., Paul, A., Göker, A., Myrhaug, H., Desruelle, H., Martin, A.: On the Design and Development of *webinos*: A Distributed Mobile Application Middleware. In: Göschka, K.M., Haridi, S. (eds.) DAIS 2012. LNCS, vol. 7272, pp. 140–147. Springer, Heidelberg (2012)
9. Open Mobile Terminal Platform organisation. Bondi website, `http://bondi.omtp.org/`
10. PhoneGap. Phonegap website, `http://www.phonegap.com`
11. Recordon, D., Reed, D.: Openid 2.0: a platform for user-centric identity management. In: Proceedings of the Second ACM Workshop on Digital Identity Management, DIM 2006, pp. 11–16. ACM, New York (2006)
12. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. In: First Workshop on Mobile Computing Systems and Applications, WMCSA 1994, pp. 85–90 (December 1994)
13. Taivalsaari, A., Mikkonen, T.: The web as an application platform: The saga continues. In: 2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), August 30-September 2, pp. 170–174 (2011)
14. Tanner, J.: Cellcos get wac on os fragmentation (March 2011)
15. The W3C. Mobile Web Application Best Practices. Technical report (2010)
16. The W3C. Widget Packaging and XML Configuration. Technical report (2011)
17. The webinos Consortium. The webinos consortium website (2012), `http://webinos.org`

Social Interaction in Gamebased Applications on Smartphones in the Context of Tourism

Dominik Grüntjens, Gerrit Lochmann, Johannes Siebel, and Stefan Müller

Universität Koblenz-Landau, Insitute for Computational Visualistics, Koblenz, Germany

Abstract. We present and compare two different approaches for touristic applications using smartphones. Our goal is to add value to the touristic experience in an appropriate way by provoking or improving social interaction between tourists. Because touristic actions always are motivated intrinsically, we decided to implement two game-based approaches. We use smartphones in two completely different ways: In the first approach, we use it as an input device for a large interactive display which is exhibited in public. In our second approach, we use it to enable tourists to explore places all over the world in a long term multiplayer game.

Keywords: tourism, gamebased, smartphones.

1 Introduction and Related Work

Nearly every tourist owns a smartphone and uses it to discover new sites ([21]). Our goal is to use smartphones to improve social interaction between tourists. Social interaction can be a conversation, a common gaming experience or the overcoming of situations in which tourists are confronted with others or in which they can play together. One trait that all situations that lead to social interaction have in common is that tourists *communicate* with other tourists. If such systems provide fun and improve social interaction, they are more likely used by tourists. In our oppinion, social interaction adds value to the touristic experience. In tourism, actions have to be motivated completely intrinsically (see [17]). Thus, we chose two gamebased approaches.

Many projects implement mobile touristic games. In [13], [3] and [22], location-based interactive stories are implemented and evaluated. In [14], a mobile tourist guide was implemented evaluating the requirements of tourists. Finally, in [12], a game with social aspects was implemented and evaluated. Services like Foursquare, Google Places, Wikitude or Layar also show that there is a demand for location based applications.

Stationary systems are also common in tourism. In [5] and [9], the possibilities of mixed reality systems in museums were discussed. Another project named 'One Rock' is very informative about social interaction while using Augmented Reality telescopes in public places [15]. In [6], another telescope providing game-based interaction and exploration for tourists was implemented.

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 400–411, 2013.

There are also works connecting a mobile phone controller with a large public display. In [18], it is shown how cell phones can be used to interact with a large public display. In [19], cell phones were used as controllers for a racing game on a large public display. Finally, in [1], a smartphone application is connected to stationary systems: The app leads to Kiosk systems in a city. Their system, 'Smartymote', also had a social component.

2 Social Interaction in Smartphone Games

One opportunity to enhance touristic experience is to enable social interaction between tourists. There is plenty of social interaction in games. Some games encourage social interaction by long term motivation, while others encourage social interaction by attractive, fast and easy experiences with others. Social interaction motivates humans and adds value to their experiences. Apps in a touristic context can only be motivated intrinsically and not extrinsically: 'There is no need to learn or work when people travel around the world - tourists want to spare free time in their holidays and not to work or to learn' ([17]). In ([8]), it is stated that 'Consumer researchers argue that the 'experiential' aspects of consumption, like consumer fantasies, feelings, and fun, play an important role in consumer choice behavior.'

We studied two different opportunities for social interaction: First, all players interact at one place. Second, players are spread all over the world while playing the same game. Before the tests we asked our test persons which way of playing games they prefer. 78% answered they like playing with others using a single display, while only 50% like playing games while being at different places. People prefer to interact with other players directly than computer mediated like in online games.

Generally, stationary consoles are not used in public places or in a touristic context today. In contrast, portable consoles or smartphones are more commonly played within public spaces because they are much more flexible. Players can decide to play the game simultaneously when they meet or asynchronously at different places. To make a statement about the richness of social interaction comparing these two approaches, we implemented two games using smartphones: A mobile asynchronous strategy game and a stationary synchronous action-game played on a public display using smartphones as game controllers.

3 Slingshot: A Stationary Mutliplayer Game with Smartphone I/O

This approach enables tourists to play a console-like game in public using their smartphones. A large public display attracts the attention. Tourists can download a special Smartphone-Controller application. The user input happens by moving the smartphone like a Wii-Controller and using the touchscreen for buttons and drag and drop gestures. The main action of the game takes place on

the large display that is visible to all players as well as spectators. Only some private details like special items are shown on the smartphone displays.

The concept was inspired by native smartphone games like Angry Birds or Graviturn as well as console games using mimetic interfaces like the Wii game Boom Blox. The public screen shows a virtual shooting range scene while the smartphone touchscreens of all participants display an interactive slingshot providing a draggable animated rubber strip. As the taut rubber strip is released, a projectile is shot into the 3D scene on the public screen. The downrage is located by the pitch and yaw measurements of the smartphone. A ballistic curve ensures a synchronous feedback to the player's movements and helps to predict the trajectory of the projectile.

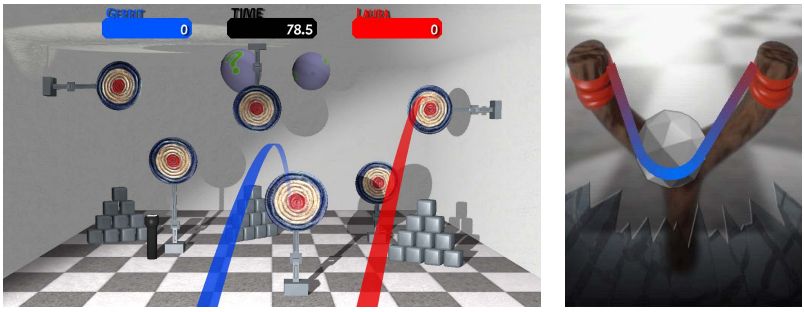


Fig. 1. Left: The app on the public display. Right: The slingshot on the smartphone's touchscreen.

Up to four players are able to play the game simultaneously. During a timespan of 120 seconds they try to hit as many targets as possible and hamper other players by shooting projectiles with special competitive effects: A 'curse item' reduces the range of the opponent's ballistic curves while the 'dynamite item' detonates after a short delay, hits many targets at a time but can be eliminated previously by the opponents.

4 Treasures: A GPS-Based Mobile-Trading-Card-Game

The second approach presented in this paper is a mobile trading-card-game. There are several major differences between the two game concepts in the way they engage the player. The two most notable ones are that Treasures is not limited to certain locations and that the sessions of Treasures are – on average – shorter than Slinghot's sessions. The player is permanently provided with surrounding venues based on the smartphone's geo coordinates (similar to Foursquare). These venues can be used to perform game-related actions, like searching for a duel or checking in. Treasures aims to improve the traditional Check-In System that is widely used by many popular applications such as

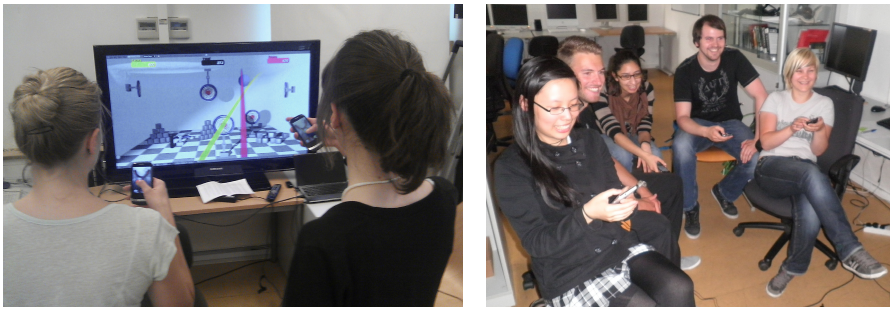


Fig. 2. Left: Two players using their smartphones to interact with the game. Right: A group of students enjoying Slingshot.

Foursquare, Facebook, and Google Places. For this purpose the game introduces common game design patterns of social games like grinding and asynchronous interaction between players. The usage of said patterns strives to give the social interaction between the players a deeper meaning, thus motivating the users to play the game on a long-term basis.

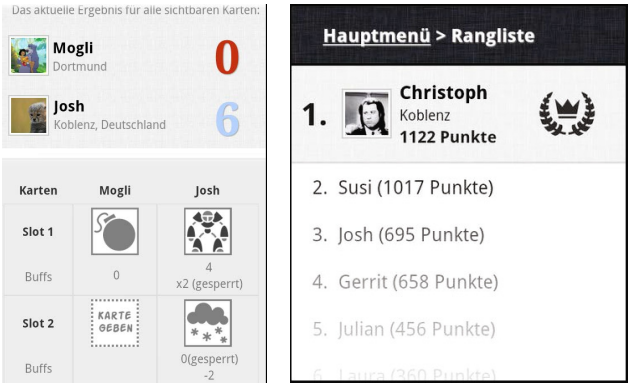


Fig. 3. Left: The duel screen. Right: The highscore screen.

The game is inspired by card-games like Magic: The Gathering and Munchkin. Some rule-based accommodations were carried out to adapt to the mobile-game nature (e.g. the game's duels are not turn-based but asynchronous). Players check in at a venue and receive game cards and points as a reward in return. They can face other players in duels when they search for a duel at the same location. Each player can use up to six game cards in a duel to achieve a score higher than the opponent's score. Additionally, other players can interfere in a duel by offering cards to the participants of a duel.

On top of that each user is assigned to one of two fractions depending on his home city and current location. The fractions are "Tourists" and "Residents".

In the current prototype duels can only take place between a tourist and a resident. The fractions were introduced to test the influence of a loosely tied group system on the social interaction between the players. By giving players who do not know each other a common ground, they get a new reason to interact with each other.

Finally, there are different goals to the game that a player can chose depending on his or her preferences. A player can try to conquer venues for the assigned fraction by winning duels at a location and thus, incrementing the score of the own fraction at this venue. The team with the better score is considered the owner of a venue. This allows players to defend their home town against tourists and vice versa, it also allows invading foreign cities. Furthermore, a player can also try to improve his own score to achieve a better position in the player highscore list. The score of a player depends on the number of check-ins, the performance in duels and the help offered to other players in duels (see figure 3). Thus, a player will constantly be rewarded for interacting with his environment.

5 Evaluating Social Interaction in the Two Approaches

We evaluated two independent groups of test persons. There have been 11 test persons for Treasures (3 female and 8 male) with an average age of 25.5 years. For Slingshot, we evaluated 23 persons (6 female and 17 male) with an average age of 25.6 years.

We decided to restrict the evaluation process as little as possible. We evaluated Slingshot by running the game on a large TV. The test persons could play the game completely without any restrictions. They could form groups to play in, they could just watch or play by themselves. On the other hand, Treasures has been played completely free for a week by the test persons.

We created a questionnaire to evaluate the person's social interaction during the game which we presented to both groups of test persons after they played. The questionnaire consisted of different questions regarding the test persons attitude towards multiplayer games, their player type and the social interaction in the game they played. The questionnaire aimed to evaluate social interaction.

In average, our test persons play computer games between 4 and 5 hours per week. 72% of the test persons for Treasures like playing computer games, and 56% of the test persons for Slingshot like playing computer games. Thus, our test persons had a positive attitude towards computer games. We assume that most tourists generally like computer games but also are not hardcore gamers either.

For a better categorization of the results, we evaluated which type of player our test persons are. According to Richard Bartle [2], there are four different player types: Achiever, Explorer, Socialiser and Killer. Achievers 'give themselves game-related goals, and vigorously set out to achieve them' [2]. Explorers try to explore the virtual world of a game. Socialisers 'use the game's communicative facilities' [2]. Finally, killers aim to 'cause distress to other players' [2].

The player type strongly influences how a player is engaged by a game. Each player was asked to rate his or her affiliation to each Bartle type on a scale from zero to four. Figure 4 depicts the numerical average of said rating. In general, the differences between the participants in the polled groups were not significant, only the Socializers were notably less represented in the Slingshot group than in the Treasures group.

Since both games rather focus on competition between players, the lack of Socializers in the participant group does not have noteworthy consequences for the evaluation. However, with the Explorer being the most prevalent player type and with both games only slightly utilizing explorational elements, effects on the perception of the games cannot be conclusively ruled out. Both games intensively utilize elements that rather appeal to Killers and Achievers, thus the game might appeal to players that also have traits of these player types.

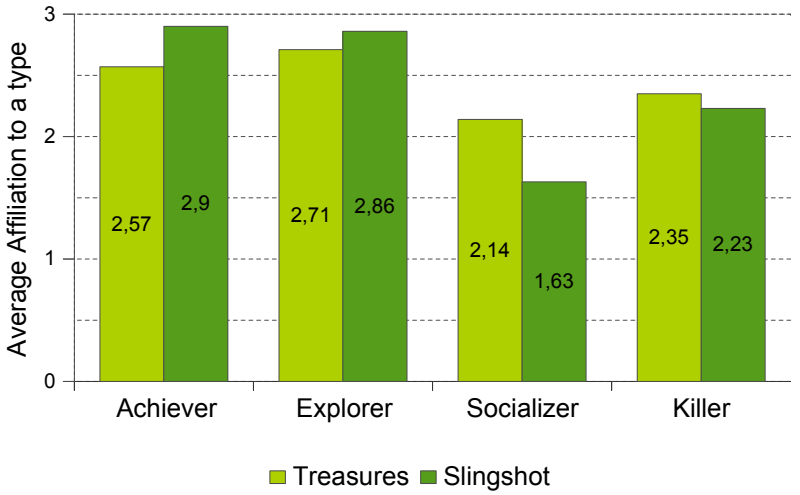


Fig. 4. Self-assesment of the participants of the second evaluation regarding their player type

5.1 Attitude towards Social Interaction

To evaluate if social interaction in the context of games plays an important role, we asked general questions to all 35 test persons (25 male and 9 female) of Slingshot and Treasures in a previous survey about their attitude towards multiplayer games. 63% stated that they like playing competitive games while 92% enjoy playing cooperative games with other human players. 71% enjoy playing games where teams act against each other. These figures show that particularly games are favored in which players have joint goals. Such games require a high amount of social interaction. Although working together requires communication among the participants, only 38% of the test persons see the need of direct communication like face-to-face communication. In addition, social interaction

goes beyond the duration of a game round if players talk about the game while not playing it. 76% of the test persons stated that they talk about computer games while not playing. For example this is the case when players exchange strategies or share their game impressions afterwards.

5.2 Consolidation of Existing Acquaintances

The following items aimed to evaluate the applications' opportunities to consolidate existing acquaintances between players. Figure 5 shows the results.

1. Because of the game, I talked to other players in general.
2. I prefer playing with friends rather than with strangers.
3. I would like to play the game with other people again.
4. The game is an interesting group activity.

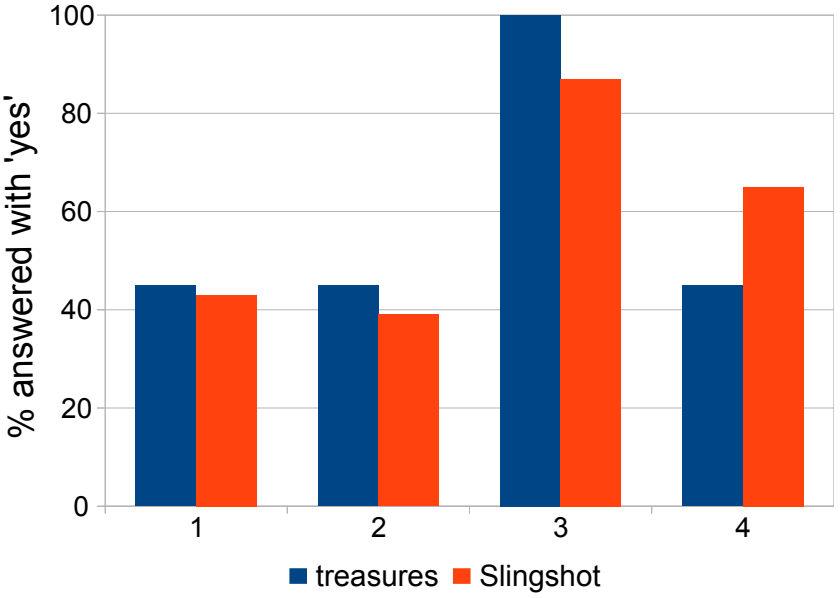


Fig. 5. Results for the items in the section 'consolidation of existing acquaintances'

Over 40% of the players talked to each other in general because of the two games (question 1, Treasures: 45%, Slingshot: 43%). 45% of the players in Treasures and 39% of the players of Slingshot stated that they preferred playing the game with friends rather than strangers (question 2). This means that many of the test persons would play the games also with strangers. Most test persons wanted to play the games again with other people (question 3, 100% Treasures and 87% Slingshot). Players liked both games and would not only like to play them

again but also to play them again with other people. Slingshot seems to be a more attractive group activity (65% Slingshot, 45% Treasures). We expected this result because the game concept of Slingshot aims to be a dedicated group activity.

5.3 Making New Acquaintances

The next item group aims to evaluate if the games can improve socializing. It can add value to touristic activities to make new acquaintances.

5. I would play with strangers.
6. I got to know new players by playing the game.
7. One can get to know new people with this game.
8. In a foreign place, I could get to know new People with this game.
9. It doesn't bother me if strangers watch me playing.
10. I prefer playing with friends rather than with strangers in public.

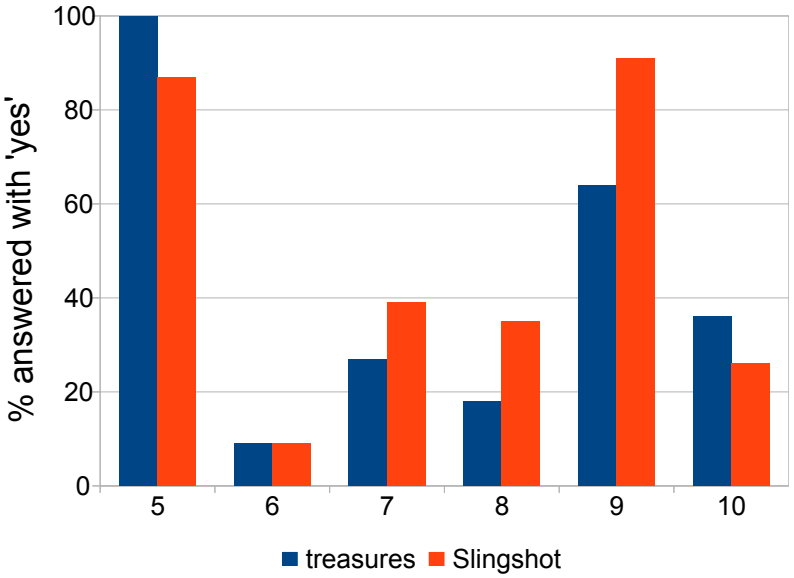


Fig. 6. Results for the items in the section 'making new acquaintances'

Nearly all test persons would play the games also with strangers (question 5, 100% Treasures, 87% Slingshot). Slingshot has a slightly lower result because the game experience is more personal. You have to actively play with others players while Treasures can be played without meeting other players. Most test persons didn't make new acquaintances while playing the games (question 6, 9% of the test persons) but many stated that it would be possible to make new

acquaintances with the games (question 7, 27% Treasures, 39% Slingshot). The fact that test persons stated that Slingshot is the better game to make new acquaintances was expected by us because it forces the players to play at one location while this is only optional in Treasures. Secondly, Slingshot uses a large public display. This enables people who are not participating to watch the game. The result is slightly lower if it is not only about to make new acquaintances but to make them in a foreign place (question 8, 18% Treasures, 35% Slingshot), but still Slingshot is better to make new acquaintances. Most test persons wouldn't be bothered if strangers watched them playing (question 9): 64% Treasures, 91% Slingshot. People would be more bothered being watched playing on their smartphone than playing on the public display. Players expect others to watch the gaming activity on a public screen, but they do not expect them to watch them doing something on their smartphone. Many people would play the games with strangers in public. Only 36% would prefer playing Treasures with their friends and 26% would prefer playing Slingshot in public with their friends instead of playing it with strangers (question 10).

5.4 Social Dynamics

In the last part of the evaluation of social interaction, we aimed to find out how far the two games can provoke and improve social dynamics between the players.

11. I talked with other players about the game.
12. I distressed other people in the game.
13. I helped other players in the game.
14. I could introduce other players into the rules.
15. Others distressed me in this game.
16. Other players helped me understanding the rules.
17. I felt mischievousness while playing.
18. I preferred the game as a single player game.

Both games made most of the test persons talk to each others about the game (question 11, 82% Treasures, 83% Slingshot). Players teased other players in both games (question 12, 91% Treasures, 65% Slingshot). In Treasures, players teased each others more than in Slingshot due to the fact that it is a long term game in which they can find new strategies to tease others. Slingshot is a more spontaneous experience where players don't tend to tease each other as much. While many test persons distressed other players, they didn't help each others in the games as much (question 13): 36% of the players helped other players in Treasures, while only 13% helped others in Slingshot because both games are mainly competitive. Most test persons stated that they could introduce other players into the games (question 14, 82% Treasures, 91% Slingshot). All players felt that they have been teased by other players in Treasures (question 15), while 57% in Slingshot felt this way. The action oriented game play of Slingshot makes players try to get as much points as possible. For this, it is not necessary to tease other players. In Treasures, it is necessary that players tease others due

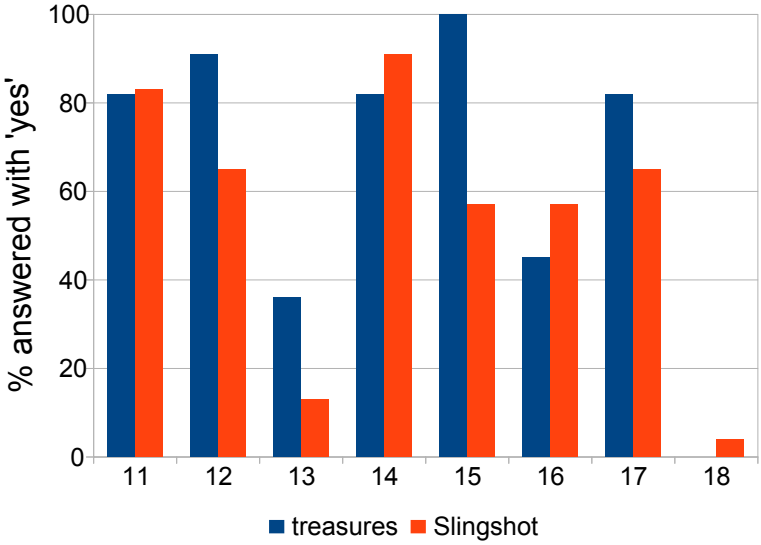


Fig. 7. Results for the items in the section 'social dynamics'

to the game concept. 45% of the players in Treasures and 57% of the players of Slingshot felt that others helped them to understand the rules of game (question 16). There were many players who felt mischievousness while playing (question 17, 82% Treasures, 65% Slingshot). Once again, it is shown that teasing others is necessary in treausures but not in Slingshot. Both games improve social dynamics by mischievousness. Nearly nobody would prefer the games as single player games (0% Treasures, 4% Slingshot).

6 Conclusion and Prospect

Both games initiated significant social interactions. In Slingshot, social interaction is generated by a fast, action oriented group activity. Players interact directly with each other by interrupting or helping other players. Games like Slingshot are great for tourism because people can play them using their own smartphones. In addition to that, such systems are interesting because they are not widely used yet. Social interaction in Treasures is rather indirect. Players do not have to meet up to play the game. Furthermore, there is a strong competition between players.

Games like Slingshot and Treasures could also be connected to create a system in which players can play with each other at all times and where they can also meet up to play at stationary public displays. Such a unified mobile app could lead them (in a playful way) to interesting venues which are equipped with stationary public displays. This idea fits perfectly to the application area of

tourism: Players could not only play such a game when they are on vacation but also in their all-day-life. Thus, this would enable tourists to stay in touch with their friends on the one hand while also regularly giving them information about venues nearby on the other hand.

References

1. Bae, M.: Smartymote: Revolutionizing the Gaming and Social Experience (2011)
2. Bartle, R.A.: Hearts, Clubs, Diamonds, Spades: Players Who Suit Muds (1996)
3. Bell, M., Chalmers, M., Barkhuus, L., Hall, M., Sherwood, S., Tennent, P., Brown, B.: Interweaving mobile games with everyday life. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (2006)
4. Brenner, C., Paelke, V., Haunert, J., Ripperda, N.: The geoscope - a mixed-reality system for planning and public participation. In: Proc. of the 25th Urban Data Management Symposium (2006)
5. Fraser, M., Bowers, J., Brundell, P., O'Malley, C., Reeves, S., Benford, S., Ciolfi, L., Ferris, K., Gallagher, P., Hall, T., Bannon, L., Taxén, G., Hellström, S.O.: Retracing the past: Mixing realities in museum settings. In: Proceedings of Conference on Advances in Computer Entertainment, ACE (2004)
6. Grüntjens, D., Arndt, D., Beschorner, J., Dietterle, J., Epe, J., Kobold, P., Stüttem, T., Müller, S.: telARscope: Gamebased Concepts for a Touristical Augmented Reality Telescope. In: GRAPP (2012)
7. Gustafsson, A., Bichard, J., Brunnberg, L., Juhlin, O., Combetto, M.: Believable environments - Generating interactive storytelling in vast location-based pervasive games (2006)
8. Goossens, C.: Tourism Information and Pleasure Motivation (2000)
9. Hindmarsh, J., Heath, C., vom Lehn, D., Cleverly, J.: Creating assemblies: aboard the ghost ship. In: Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work, CSCW 2002, New York, USA, pp. 156–165 (2002)
10. Juraskope. Juraskope project (2007), `http://www.artcom.de/projekte/projekt/detail/juraskope/`
11. Lutz, B., Becker, M., Stricker, D., Bockholt, U.: The augmented reality. In: ACM SIGGRAPH International Conference on Virtual Reality Continuum and its Applications in Industry, pp. 352–354 (2004)
12. Nikkila, S., Linn, S., Sundaram, H., Kelliher, A.: Playing in Taskville: Designing a Social Game for the Workplace. In: Workshop on Gamification Using Game Design Elements in NonGaming Contexts, CHI 2011 (2011)
13. Paay, J., Kjeldskov, J., Christensen, A., Ibsen, A., Jensen, D., Nielsen, G., Vutborg, R.: Location-based Storytelling in the Urban Environment (2008)
14. Rasinger, J., Fuchs, M., Beer, T., Hopken, W.: Building a Mobile Tourist Guide based on Tourists' On-Site Information Needs. In: Tourism Analysis, vol. 14 (2009)
15. Reeves, S., Fraser, M., Schnädelbach, H., O'Malley, C., Benford, S.: Engaging augmented reality in public places. In: Adjunct Proceedings of SIGCHI Conference on Human Factors in Computing Systems (2005)
16. Schnädelbach, H., Koleva, B., Flintham, M., Fraser, M., Izadi, S., Foster, M., Benford, S., Greenhalgh, C., Rodden, T.: The augurscope: A mixed reality interface for outdoors. In: Proc. CHI 2002, pp. 9–16. ACM Press (2001)
17. Stier, M., Grüntjens, D.: Factors for knowledge transfer in mobile gamebased city tours. In: Virtual and Augmented Reality in Education (2011)

18. Thoresson, J.: PhotoPhone entertainment. In: CHI 2003 Extended Abstracts on Human factors in Computing Systems (2003)
19. Vajk, T., Coulton, P., Bamford, W., Edwards, R.: Using a Mobile Phone as a Wii-like controller for Playing Games on a Large Public Display. International Journal of Computer Games Technology (2008)
20. Verkasalo, H.: Empirical Observations on the Emergence of Mobile Multimedia (2006)
21. Wang, D., Park, S., Fesenmaier, D.R.: An Examination of Information Services and Smartphone Applications (2011)
22. You, Y., Chin, T.J., Lim, J.H., Chevallet, J.-P., Céline, C., Nigay, L.: Deploying and evaluating a mixed reality mobile treasure hunt: Snap2play. In: Henri ter Hofte, G., Mulder, I., de Ruyter, B.E.R. (eds.) Mobile HCI. ACM International Conference Proceeding Series, pp. 335–338. ACM (2008)

Enhancing Traveler Context through Transferable Activity Patterns

Chad A. Williams[1], Abolfazl Mohammadian[2], Joshua Auld[3], and Sean T. Doherty[4]

[1] Department of Computer Science
Central Connecticut State University, New Britain CT, USA
[2] Dept. of Civil and Materials Engineering
University of Illinois at Chicago, Chicago IL, USA
[3] Argonne National Laboratory, Chicago IL, USA
[4] Dept. of Geography and Environmental Studies
Wilfrid Laurier University, Waterloo, Ontario, Canada

Abstract. Developing a model of the needs of a mobile traveler is critical to good personalization. Transportation planners have been modeling these needs for years, but these models have not been used to date due to two outstanding questions: 1) are they applicable to individual travelers 2) are they useful beyond the studied region. This study demonstrates these studies can directly enhance the model of mobile users, and be done in a practical way through the transference of activity patterns across cities. This work then demonstrates how these studies can be combined with patterns of an individual mobile user successfully.

Keywords: travel prediction, activity prediction, traveler context.

1 Introduction

With the increase in hand held computers and GPS devices, there has been an increasing demand for predicting an individual's future travel plans for devices such as smart traveler's assistants. Most existing studies of projecting the patterns of individuals have examined learning traveler behavior by concentrating on the travel itself or location attributes such as Krumm and Horvitz [1]. Other approaches have examined trying to determine the activities of the person based on location such as that by Liao et al. [2]. More recent work by Zheng et al. has tried to combine these approaches by using GPS traces for both activity and location recommendation [3]. These works and others have shown that a considerable amount of information can be inferred about a traveler based solely on analysis of their GPS trace. This study takes a different approach by instead looking at the decision factors and reasons behind the travel.

Examining the reason behind travel has been studied by transportation planners through detailed, time-consuming surveys conducted to capture not just travel, but the planning and decision processes that go into travel. Unfortunately these data collection methods are too burdensome on users to be practical for

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 412–423, 2013.

mobile applications. In this work we examine using information gathered in these data-rich surveys to enhance the model of mobile users with limited user input. Specifically the aim of this work is to use data that can be collected from a mobile user unobtrusively such as GPS analysis and augment that data without burdening the traveler.

In this work, we address two questions. First, can travel surveys be used to enhance the patterns of mobile users in locations other than the study area? While it is somewhat intuitive that this might work if the survey area is the same as that of the traveler; being restricted to the areas where a survey was conducted would greatly reduce the applicability of this approach. To address this question this study examines how well the patterns of one city might be used in a different metropolitan area. Second, we examine how to augment the observed travel characteristics with patterns observed in these surveys.

One of the goals of this study is to use an understandable model such that the factors influencing a particular outcome can be readily understood. The reason for this criterion is that an easily understood model is likely to make it easier to create personalization that could influence the actions of the traveler. Prior work has shown that sequential associative rules are particularly effective for traveler context prediction using survey data [4]. However for the problem described above there is significantly more missing data compared to the survey prediction problem. As a result, the second aspect we examine is a comparison of traditional sequential rules with attribute constrained rules (ACR) which were specifically created to address sequential mining with missing data [5]. As this work will empirically demonstrate, ACRs can be used to learn the patterns of an individual and augment them effectively with general patterns outside of the original survey area despite large amounts of missing data.

Below we begin by introducing the problem and motivation for this study, followed by a formal description of the problem and the methods used. Next, an empirical evaluation of transference and the appropriateness of the ACR learning method are demonstrated. Finally, a discussion of the impact of this work and future areas of study are presented.

2 Background and Motivation

Transportation planners have studied travel behavior extensively over the years. More recently, their focus has shifted from looking at travel alone to understanding why a trip is made and when this decision was made [6]. One of the approaches used for this has been examining the activity needs of the person as the reason the travel is made. This is important because by understanding the activity priorities of travelers, not only can a model be built to project what activities a person is likely to do over a fixed period of time; a model of what influences their behavior can also be obtained. We propose this type of information is more important than travel and location information alone in understanding what information is useful to the traveler before they *plan* their trip.

The inputs for these models are collected through activity-based travel surveys. These surveys consist of a person meticulously recording all of their trips, important locations, activities at every stop, and reasons these were made. Some go a step further and have participants record planning information for when an activity, location, and schedule were determined. Finally, planners take the information gathered from the surveys to build activity models which are then used to project travel patterns for people in the area of study. In creating these models, a key assumption is made that activity models can represent the basic patterns of people within a similar area. The validity of this approach has been demonstrated when applied to the **same** metropolitan area [7].

While this type of approach has been verified within the same metropolitan area at an aggregate level, the ability to use activity patterns within one city to project the activity patterns in another metropolitan area at an individual level has been largely unexplored. We assert that since activity needs are the factor driving travel, similar activity patterns will be seen regardless of the environment. Thus, we propose patterns from one metropolitan area can be transferred to other metropolitan areas. This is important because without this transferability, only cities where an activity survey has been conducted would benefit from this information. While many cities do conduct some form of activity surveys, the information collected can vary greatly making it impractical to apply location specific studies to every mobile user. If patterns from one city could be shown to be transferable to other cities, this would mean a general model could be applied to wherever a user might be.

The basic concept behind activity-based analysis is that all travel is driven by the activity needs of the individual. Research has shown the planning of these activities happens in a fluid manner based on both personal flexibility and activity flexibility, making these aspects critical for planning decisions. As a result, the goal of traveler prediction is not activity or location prediction, but traveler context prediction which includes these flexibility and planning factors.

The contribution of this paper is demonstrating that the micro patterns of one city can be used to augment the observations of travelers within a different city for more complete modeling of the trip characteristics of an individual traveler. In the past the amount of data entry collected from the user to accomplish this would not be practical; but with GPS enabled smart phones a significant amount of detail about the trip can be captured passively making the task of inferring additional information much more tractable. While this work does not explore if this relationship holds across significant cultural differences, it does show that these patterns can be transferred across varying transportation networks and urban structures within a selection of cities within the United States and Canada.

3 Partially Labeled Sequence Completion

In our analysis, we break the trips into discrete steps of travel and activities. For each of these discrete steps there are a number of different characteristics associated with each segment such as trip time, mode of travel, activity, and location.

Here we address adding to those characteristics information such as personal and activity flexibility information, as well as when various aspects of these were decided. When put together these form a sequence of sets of activity and trip characteristics describing that user's travel.

Algorithms for rule mining of sequential patterns have been a major source of interest since they were first introduced in Agrawal and Srikant [8]. The problem we examine is given a sequence where there are a known set of attributes that describe an event within the sequence, infer any missing values of the attributes for a target set. This problem addresses the situation of knowing a set of attributes through methods such as a GPS trace analysis, but needing to infer missing flexibility information about a traveler.

Problem Statement. For partially labeled sequence completion, let

$$H = \{H_1, H_2, \cdots, H_n\}$$

be a database of sequences, and let:

$$H_i =< S_1, S_2, \cdots, S_n >$$

be the sequence of sets of observations in a sequence i; where each observation set S_j is composed of 1 to m attributes $\{a_{j,1}, a_{j,2}, \cdots, a_{j,m}\}$. Each attribute $a_{j,k}$ has a discrete set of values for the k^{th} position that is shared across all observation sets S. Intuitively the sequence H_i can be thought of as a series of traveler contexts (sets) with discrete measures (the attributes), where at each event j all measurements are relevant, but only a portion of these measures may actually be recorded in the set S_j. Given a sequence H_{target} of length l and a target set S_t to be completed where $1 \leq t \leq l$ and between 1 to m arbitrary attributes are missing values. Determine the values of all missing attributes $\{a_{t,1}, a_{t,2}, \cdots, a_{t,m}\}$ in S_t. Thus our goal is to use the surrounding sequence information in H_{target} to populate any missing values to complete the set S_t. In other words, given some information that can be determined about a traveler from GPS analysis, can we infer the missing traveler context information.

3.1 Sequential Set and Attribute Constrained Rules

In this section we discuss how the associative sequence mining and prediction problem relates to the traveler context prediction problem. This is followed by an illustrative example of how ACRs introduced in Williams et al. are better suited for this problem since missing data is an issue [5]. This approach stems from the general apriori associative sequence prediction problem, but is a specialized form of it more suited to the specific traits of this problem. A comparison of the results of the two will be discussed in the results section.

One characteristic of the problem being examined is that unlike the general form of sequential set prediction, the travel context prediction problem is constrained to always contain the same set of attributes describing the travel. This sub-problem of sequences of sets referred to as partially labeled sequences,

applies to sequences that have sets of a fixed group of attributes where some attribute values (labels) may be unknown. Due to this structure it is also possible to tell which attributes are missing rather than being a negative example that the attribute does not have the specified value. For this sub-problem ACRs were introduced to use these types of constraints to better estimate the benefit of a sequential rule when missing data is an issue [5]. Specifically ACRs work by identifying templates of sequential patterns to mine rules that specifically target completions of particular attribute values when missing values are an issue.

3.2 Illustrative Example

Here we provide an illustrative example to demonstrate the benefit of ACRs when missing data is an issue. Throughout these examples refer to Table 1 as the sequence database. Below we use the standard definitions of *support* and *confidence* defined as: The **support** of the sequential rule $X \rightarrow Y$ is the fraction of sequences in the database that contain Y. The **confidence** of a sequential rule $X \rightarrow Y$ is the fraction of sequences in the database that contain X that also contain Y.

Table 1. Example sequence database

H_1	$< \{a_1\}\{b_1\}\{a_2, b_2\} >$
H_2	$< \{a_1\}\{a_2, b_1\}\{a_2, b_2\} >$
H_3	$< \{a_1\}\{b_1\}\{a_2\}\{b_2, c_2\} >$
H_4	$< \{a_1\}\{a_2, c_1\}\{b_1\} >$

For an example of how constrained rules can better represent the applicable confidence, consider the following scenario: $H_{target} = \langle\{a_1, b_1\}\{b_1\}\{a_2,\ \mathbf{?}\}\rangle$, where the last set, $S_3 = \{a_2,\ \mathbf{?}\}$, is the target set and we are interested in completing the set with the value of attribute b. The following traditional sequence associative rule would be applicable:

$$\langle\{a_1\}\{b_1\}\{a_2\}\rangle \rightarrow \langle\{a_1\}\{b_1\}\{a_2, \mathrm{b}_2\}\rangle$$
$$[\mathrm{sup} = 2/4, \mathrm{conf} = 2/3]$$

Where S_3 can be completed $\{a_2, b_2\}$ with a confidence of 2/3.

With ACR this idea is extended to constrain pattern matches to particular attribute values of interest. In our example, since we are specifically interested in the value of attribute b, the ACR version of the same rule would be:

$$\langle\{a_1\}\{b_1\}\{a_2, \mathbf{b{:}*}\}\rangle \rightarrow \langle\{a_1\}\{b_1\}\{a_2, \boldsymbol{b}_2\}\rangle$$
$$[\mathrm{sup} = 2/4, \mathbf{conf = 2/2}]$$

where $\{a_2, \mathbf{b{:}*}\}$ is a template such that matching examples in the history must contain a_2 and **some** value for b, in other words the attribute value can't be

missing. This results in the calculation of rule confidence only considering sequences that could either support or negate the consequence rather than being counted as a negative example because a value is missing. Thus, it is able to discount sequence H_3 included by sequential rules as it does not match the attribute constrained pattern. This advantage in accurately evaluating the value of the constrained sequence rule is the reason we examine ACR.

4 Evaluation

The basic idea behind activity pattern transferability is that the patterns from one metropolitan area can be used to predict the patterns for another area. The main purpose of these experiments is to compare how survey data from one city can be used to predict the activity behavior in a completely different city. To evaluate this hypothesis, a series of experiments was conducted to examine multiple aspects of the transferability of activity patterns across a selection of cities of various sizes and locations. First, we demonstrate that the patterns of one city can be transferred to another city with reasonable performance as compared to using the patterns of the original city. Next, we examine using this idea to improve learning the patterns of an individual outside of where the patterns were collected to demonstrate that this idea can be applied in general to new locations. The details of the experiment design and results are given below.

4.1 Data Selection

For evaluating this problem we chose a selection of sets of activity and planning data that were collected by transportation planning activity based surveys. The reason for this choice was in part due to the proven record of this type of data being effective at modeling traveler behavior [7]. The second reason these activity surveys were chosen was because they focus on a mix of fields that require active collection as well as fields that may be determined via passive means.

The differentiating factor between these two types of data is that *actively* collected data requires a user to enter the information to be determined. For example **why** they chose a particular location. *Passively* collected data, on the other hand, is data that may be determined without user interaction. An example of this would be the user's location since this can be determined via the GPS device. The importance of selecting surveys that combine these two factors is that while some of the actively collected data would be very useful for personalization it would likely be too burdensome to ask a mobile user repeatedly. However as this study will show by combining this data with passive means it is possible to match the patterns of the passive data and infer the active data. This would thus allow a mobile application to infer a preference profile of a user that would otherwise be impractical to collect due to the burden on the user.

Data. To make an evaluation of how well activity patterns transfer across cities, activity surveys from a variety of areas were selected. Specifically the data from the Toronto CHASE survey [9], the 2001 Atlanta Household Travel Survey [10], the 2002 Anchorage Household Travel Survey [11], and the Chicago UTRACS survey [12] were used. These data sources were selected as they provide a good mix of metropolitan populations and differences in urban structure.

For this study the following fields were compared: location, activity type (22 categories), mode, observed sequence of the activity, arrival time, departure time, duration, individuals involved (related to involvement of others in trip), whether the activity was mandatory, when planned, when decided on the location, when decided who would participate, when decided timing of activity, when travel mode selected, why travel mode selected, why route was chosen, duration flexibility, time flexibility, spatial flexibility, and when the activity was planned.

Data Summary. Table 2 summarizes the different characteristics of the data sources listed above. As this table shows, the more time consuming planning surveys (CHASE and UTRACS) tend to be considerably smaller than the basic activity surveys, but they also offer longer survey periods providing better examples of multi-day activity patterns.

Table 2. Data set characteristics

Data set	Activity Information	Planning Information	GPS Trace	Time Period	Households
CHASE 2002-03	X	X		7 days	271
Atlanta 2001-02	X			2 days	8,069
Anchorage 2002	X			1 day	1,293
UTRACS 2009-10	X	X	X	14 days	100

4.2 Methods

Within these experiments, the primary goal was to determine if activity patterns from one region can be used in a different region compared to a model built on the same city. The approach for verifying this transferability is similar to that used in Arentze et al. where the data of one city is used as training data for a second city and the results of that test being compared to the results of a city being trained on its own data [13]. For their study the evaluation was made at an aggregate level to compare distributions, by contrast here we analyze the transferability of patterns at an individual level.

For these experiments the results of training and testing for the same city were performed using a ten times cross-folding methodology. The cross-folding methodology was executed such that each fold had the same number of travelers although the number of activities per fold could be different depending on the number of activities for the users selected for a particular fold. For all other tests of training based on one city and testing on a second city were carried out by using the entire survey set of the training and target cities.

The method for executing the tests was to train the classifier and then for the test data set evaluate each traveler's sequence separately. The training data was built by taking each user sequence in the training set and splitting the sequence by home-based tour. To capture a new tour's start being dependent on the activity last completed before the current home-based tour began, each tour after the first would start with the last two elements of the previous sequence. Thus, each training user's sequence was split such that each new sequence would begin with an out of home activity, followed by a home-based activity, followed by all activities up to and including the next home-based activity. This method was used to break up long sequences such as those in the multi-day CHASE and UTRACS data where an activity sequence could contain over 80 activity sets.

The evaluation was done so that in evaluating the test user, the activity sequence of the user was evaluated in a stepwise fashion. In other words, for the first prediction of the user there was no history to base the prediction on, but for the second prediction there would now be one set of history in the sequence. Similar to the training approach, the user's history used in testing was the current home-based tour and the previous home-based tour. This was continued through the end of the sequence where each prediction of the n^{th} step of the tour was based on $n-1$ sets in the user's current tour plus the previous home-based tour. This process was repeated for each user in the test set and the average of all of these predictions is presented.

One of the challenges of learning the activity patterns of these data sets is that all four of these contain numerous missing values from survey participant's not answering all questions. This is important as it is similar to only collecting data that can be derived from passive data such as GPS traces, yet attempting to resolve data only observed by active data entry such as that in the survey data. To verify the benefits of the ACR technique in this environment, the experiments were conducted with both standard apriori sequential mining and ACR. The results showed that ACR outperformed standard sequential rules across all experiments in terms of both precision and recall. The results of standard sequential rules are not shown here for brevity.

4.3 Experimental Evaluation

As described above, the problem of partially labeled set completion involves taking a sequence and trying to fill in or predict items within a single target set within a sequence. Since the problem of partially labeled set completion can take the form of predicting anywhere from a single item in a target set to all items in the target set, the results below reflect the average of all possible combinations of the target pattern in all possible positions for the target set. Where target pattern means: the set of attribute values in the target set that are being evaluated. Thus in the experiments below, for the target set any attribute value that is not specifically of interest as specified by the target pattern retains its original

Table 3. Activity pattern transferability

F-measure

		Test			
		UTRACS	Atlanta	Anchorage	CHASE
Training	UTRACS	**0.461126**	0.447509	0.427923	0.575487
	Atlanta	0.333476	**0.443616**	0.441377	0.60563
	Anchorage	0.41161	0.461426	**0.546549**	0.57918
	CHASE	0.276572	0.417372	0.426867	**0.625694**

value for determining matching rules. For example if the target pattern included attributes a and c ($S_T = \{a_T c_T\}$). In testing the sequence:

$$\langle \{a_1 b_2 c_1\} \{a_2 b_2 c_2\} \{a_1 b_1 c_2\} \rangle$$

If the target set was S_3 for the sequence, the test sequence would thus be:

$$H_{target} = \langle \{a_1 b_2 c_1\} \{a_2 b_2 c_2\} \{a_T b_1 c_T\} \rangle$$

In making the predictions, once all rules were identified the ranking scheme discussed in Williams et al. was applied [5]. Specifically, the rules were ranked in order by confidence, number of target productions, support, and antecedent length. The productions of the top ranked rule were then applied to the target set, the remaining matching rules were then re-ranked by finding the next highest rule whose consequent did not contradict the values previously completed. This was repeated until either all rules had been exhausted or all fields in the set being tested were completed. The values shown in the tables are based on selecting the parameters of support and confidence thresholds that yielded the maximum F-measure. The precision and recall metrics discussed are based on this configuration for the maximum F-measure.

The experiments were executed so that every city was treated as a training and test set for all other cities. Table 3 contains the F-measure results of these tests. The rows represent how well the survey of a particular city was at predicting the patterns in other cities. Likewise the columns represent how well various cities performed at predicting the patterns of that particular city. Across the diagonal marked in **bold** are the results of training and testing on the same city.

As these results in Table 3 demonstrate, it is possible in nearly all of these cases to find a secondary city which can be used to train activity patterns for the first city nearly as well as the city itself. For Atlanta there appear to be two cities that can actually slightly improve the performance over using Atlanta alone, however the difference is not statistically significant. The one exception to this was Anchorage for which there was a drop off in transferability. As will be discussed further in the recall results, part of the reason for this, discovered in further data analysis, is that the Anchorage study captured a more diverse set of observations.

In examining the precision and recall results (not shown), it was interesting to note that the UTRACS survey showed the highest precision across all other test

cities. However it also had the worst recall. This is likely due to the lengthier study being better able to capture the underlying patterns, but the small size of the study hurting the number of patterns observed. This assertion about why the recall is low is further supported by the CHASE data set which also suffered from lower recall and it is also one of the smaller studies. An interesting observation from these results is that the CHASE data set has the poorest transferability. This could warrant further study to better determine if this is potentially related to differences between preferences in the United States and Canada. Another interesting result is when the model was built from the Anchorage dataset, it resulted in the highest recall across all datasets. As noted above this is likely due to this dataset having the highest diversity in observations (activity sequences, flexibility differences), while also having a relatively large mix of missing data across all attributes. Thus when the missing data is accounted for, the diversity of the observations make it well suited for covering a wide variety of environments.

As shown above, while generally transferring patterns from one city to another produced good results, the selection of the training data could significantly impact overall performance as well as the precision versus recall mix. Even so, the implication of this is that even if a user is in a city that other histories are not available, survey data from a second city could be used instead yielding good results despite the lack of city specific data.

4.4 Combined Knowledge

In the next set of experiments a comparison was made of using a model built just on a user's individual history against one based on just the general patterns as well as a hybrid model that combines the two approaches. Since one of the goals of this work is to demonstrate this approach can be used in areas outside of these survey areas; this experiment was conducted using the Anchorage data set for the general patterns and the test set was the UTRACS data set. The UTRACS data was selected as it is the longest survey thus allowing the cumulative learning to be more apparent. The points charted are the average score of predicting all fields of the activity context for the day averaged across all users for that day

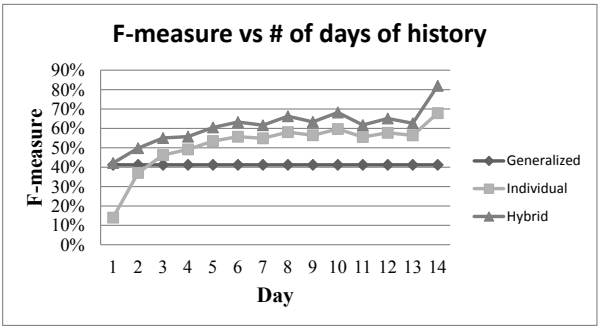


Fig. 1. Comparison of F-measure vs. patterns used

against their individual histories. In other words the points at day 3 represent the average across all users for just the patterns observed on day 3.

Using the approach described, 3 results were recorded. Figure 1 shows the comparison of the F-measure of these three different approaches day by day for the 14 day period of the UTRACS survey. First, the accuracy of predicting the individual histories using only the generalized patterns from Anchorage were recorded which is depicted as the *generalized* patterns in Figure 1. The line shown for the generalized patterns is the average across all of the days. The second, is the accuracy of predicting the patterns of the individual based solely on their own history without any generalized patterns, which is depicted as *individual* patterns. As the results show, the generalized patterns initially perform better before enough individual data is collected, which occurred at day 3. After day 3 the model based solely on the individual data performed better.

The final results are based on a combination of these two data sets depicted as *hybrid* patterns. The intuition behind creating the hybrid is to create a model of the individual augmented with general patterns to increase recall of patterns not previously seen in the user's history. The hybrid was created by using two separate training sets: the generalized patterns, and the patterns of the individual. For making predictions the predictions would be generated as complete as possible using only the individualized patterns. Once the individual patterns/rules were exhausted the generalized patterns were then used to fill in any remaining gaps. As the results show, when only that limited amount of information is provided it is difficult to learn a model of that individual quickly. This problem referred to as the "cold start" problem occurs due to limited training examples being available until enough history is gathered. As the results demonstrate not only does the hybrid benefit from the initial general patterns to avoid a cold start, it also immediately benefits from the individual patterns. By examining the results we see that the hybrid offers better performance than either method alone throughout the 14 day study period.

5 Discussion

In this paper we have examined the idea of using transportation planning activity survey data to augment personal history for a richer model of a mobile user. The goal of this work is to allow the limited data that can be collected from a mobile user through passive means such as a GPS trace to be enhanced with features such as constraints, planning and flexibility information that cannot be collected practically through unobtrusive methods. This idea of traveler context that is made up of not just current and next step activities but accessibility options, timing constraints, and scheduling behavior represents a significant increase in the likely behavior of a traveler without adding an additional burden on the mobile user. An additional benefit of the ACR technique is that the predictive model is much more transparent than techniques such as neural networks, allowing additional insight to be gained from the models created.

The goals of the majority of studies in this area have been focused on modeling what the traveler is doing or going to be doing. We propose that the end goal of the research in this area should be to build a more in depth model which includes additional detail such as planning and timing flexibility. Capturing/inferring this information provides great opportunities for personalizing travel applications before events are *planned* rather than just before the trip is made.

References

1. Krumm, J., Horvitz, E.: Predestination: Inferring Destinations from Partial Trajectories. In: Dourish, P., Friday, A. (eds.) UbiComp 2006. LNCS, vol. 4206, pp. 243–260. Springer, Heidelberg (2006)
2. Liao, L., Fox, D., Kautz, H.: Location-based activity recognition using relational Markov networks. In: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI (2005)
3. Zheng, V.W., Zheng, Y., Xie, X., Yang, Q.: Collaborative location and activity recommendations with GPS history data. In: Proceedings of the 19th International Conference on World Wide Web (WWW 2010), pp. 1029–1038. ACM (2010)
4. Williams, C.A., Mohammadian, A., Nelson, P.C., Doherty, S.T.: Mining Sequential Association Rules for Traveler Context Prediction. In: Proceedings of the First International Workshop on Computational Transportation Science held at The International Conference on Mobile and Ubiquitous Systems: Networks and Services, MOBIQUITOUS 2008 (2008)
5. Williams, C.A., Nelson, P.C., Mohammadian, A(K.): Attribute Constrained Rules for Partially Labeled Sequence Completion. In: Perner, P. (ed.) ICDM 2009. LNCS, vol. 5633, pp. 338–352. Springer, Heidelberg (2009)
6. Timmermans, H.: Progress in Activity-Based Analysis. Elsevier (2005)
7. Doherty, S.T., Mohammadian, A.: The Validity of Using Activity Type to Structure Tour-based Scheduling Models. In: Proc. of 86th Annual Meeting of the Transportation Research Board, Washington D.C. (January 2007)
8. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Yu, P.S., Chen, A.S.P. (eds.) Eleventh International Conference on Data Engineering, Taipei, Taiwan, pp. 3–14. IEEE Computer Society Press (1995)
9. Doherty, S., Nemeth, E., Roorda, M., Miller, E.: Design and Assessment of the Toronto Area Computerized Household Activity Scheduling Survey. Journal of the Transportation Research Board 1894, 140–149 (2004)
10. NuStats: 2001 Atlanta Household Travel Survey: Final Report. Technical report, Atlanta Regional Commision (April 2003)
11. NuStats: 2002 Anchorage Household Travel Survey Technical Report. Technical report, Municipality of Anchorage (September 2002)
12. Frignani, M.Z., Auld, J., Mohammadian, A., Williams, C., Nelson, P.: Urban Travel Route and Activity Choice Survey (UTRACS): An internet-based prompted recall activity travel survey using GPS data. In: Proceedings of 89th Annual Meeting of the Transportation Research Board, Washington D.C. (January 2010)
13. Arentze, T., Hofman, F., van Mourik, H., Timmermans, H.: Spatial Transferability of the Albatross Model System: Empirical Evidence from Two Case Studies. Transportation Research Record: Journal of the Transportation Research Board 1805, 1–7 (2002)

Erratum: Mobile Computing, Applications, and Services

Volume Editors

David Uhler
National Diment - Slalom Consulting
Seattle,WA 98104, USA
E-mail: davidu@slalom.com

Khanjan Mehta
The Pennsylvania State University
School of Engineering Design Technology
and Professional Programs (SEDTAPP)
University Park, PA 16802, USA
E-mail: khanjan@engr.psu.edu

Jennifer L.Wong
Stony Brook University, Computer Science
Stony Brook, NY 11794, USA
E-mail: jwong@cs.stonybrook.edu

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, 2013.

DOI 10.1007/978-3-642-36632-1_27

In the original version, the affiliation of David Uhler is incorrect. The correct affiliation is:

David Uhler
National Diment
Seattle,WA 98104, USA
E-mail: davidu@slalom.com

The original online version for this volume can be found at
http://dx.doi.org/10.1007/978-3-642-36632-1

Author Index