End-User Development for E-Government Website Content Creation

Daniela Fogli

Dipartimento di Elettronica per l'Automazione
Università degli Studi di Brescia
Via Branze 38, 25123 Brescia, Italy
fogli@ing.unibs.it

Abstract. E-government websites are currently becoming more and more huge and complex. They provide citizens with several kinds of information, including services for online task payment or front office reservation. The creation and maintenance of such websites often require a distributed approach: the content publication task is transferred from software developers to personnel of the various organization departments (here called the *publishers*). To this end, a Content Management System (CMS) is usually adopted. However, CMSs do not generally satisfy all requirements and needs that emerge in this application domain. Therefore, the adoption of End-User Development (EUD) techniques, tailored to the publishers' culture, background and skills, represents a possible solution to CMSs' current limitations. In this paper, after discussing the context and the existing problems, we describe an approach to extending CMSs with EUD techniques. The approach will be discussed with reference to the creation and maintenance of the website of an existing government agency.

Keywords: e-government website, content management system, accessibility, end user, meta-design.

1 Introduction

Government agencies are complex organizations whose websites are currently becoming more and more huge and articulate. They offer to citizens several kinds of information, including sophisticated services such as online task payment or front office reservation.

For such websites, many countries all over the world have promulgated laws to establish the duty of satisfying precise accessibility standards [23][39][53], in addition to the well-known usability requirements.

The creation of websites that satisfy usability and accessibility requirements has been traditionally accomplished by software developers. Among the other activities, software developers had to perform content authoring, by gathering information from domain experts that worked in the various departments of the government agency. However, this centralized organization was doomed to fail whenever the website contents increased considerably: software developers became a bottleneck, thus determining significant delays in the publication process.

V. Pipek et al. (Eds.): IS-EUD 2009, LNCS 5435, pp. 126–145, 2009.

For these reasons, a decentralized strategy is currently preferred in most government agencies: the responsibility of publishing contents on the website is assigned to the employees of the different agency departments. Distributed content authoring has a significant impact on the work organization and personnel roles, and investments are necessary to acquire proper software applications that allow storing, controlling, versioning, and publishing various kinds of web material. Web Content Management Systems (CMSs) are the software applications that meet this demand.

A CMS is usually installed and managed by software developers – personnel internal or external to the government agency who are expert in computer technology. In such a situation, their responsibility in web page creation is reduced with respect to the centralized approach. In particular, they have to develop page templates, by ensuring their accessibility and usability. They also have to design the navigation architecture, by assigning privileges to personnel committed to content publication, the so-called *publishers*. Publishers – personnel working in some agency department who possess the knowledge about the content to be published on the website – are the very end users of the CMS. As such, these personnel are expected to evolve from passive interviewees during requirements analysis to active content producers. They generally have limited competencies in computer technology, but may be acquainted with web browsers, word processors, spreadsheets and other similar office applications. Therefore, they are capable of using content authoring tools available in the CMS to add their contents to the website.

However, from our collaboration experience with a large Italian municipality, it emerged that CMS' end users are often required to acquire some programming skills, in order to cope with the limitations of the CMS. In particular, we have observed that existing CMSs often lack functionalities for generating HTML code that satisfies accessibility requirements as established by national laws (this is particularly true when the adopted CMS is out-to-date, but migration to another product is not manageable). In such a situation, publishers must access the HTML code generated by the CMS authoring tools, and modify it manually. Consequently, this requires to provide publishers with training courses and/or manuals that support them in doing this job. In spite of these precautions, people not expert in computer technologies keep on considering this task as difficult, error-prone and time consuming, and thus they arrive even at refusing to perform it.

Furthermore, software developers still remain a bottleneck in creating online services. In fact, after the elicitation of requirements from personnel of the different agency departments, they are in charge of implementing the services through the programming language available in (or compliant with) the CMS. Moreover, the communication gap [9][20][31] that often exists between developers and domain experts (the departments' employees) creates problems in correctly designing and developing these services. Therefore, tools are needed to support CMS' end users in creating not only static content, but also online services, since these people are the owners of the necessary domain knowledge.

We argue here that both HTML editing and online service creation must be regarded as two different kinds of software development activities publishers should carry out. We propose to achieve this goal by extending CMSs with proper End-User Development (EUD) facilities.

EUD [29][50] has been defined by EUD-Net, the network of Excellence on End-User Development funded by the European Commission during 2002-2003, as "the set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create or modify a software artifact" [16]. EUD activities can range from just setting parameters to the use of macro languages to extend system functionalities [8][9][28][35].

The contribution of this paper is to promote the application of results of EUD research in the field of CMSs, with a particular focus on the use of CMSs for creating e-government website content. With reference to the real case of an Italian municipality, we propose to integrate a CMS with tools that support publishers in creating contents and services, transparently with respect to their underlying representation. In this way, publishers can behave as "unwitting programmers" [11][41] by creating or modifying software components without being conscious of this. To this end, EUD tools must be designed by relying on publishers' competencies and skills, thus avoiding them to acquire new competencies in computer technologies.

In particular, we carried out a case study research [57][58] concerning an EUD-based technique that has been implemented to relieve publishers from HTML editing. This research is extensively described in [19] and briefly summarized here. A second technique has been investigated and discussed with software developers that work at the Italian municipality. It is still at the early stages of development, but it is presented here as very promising in the case of online service creation.

The paper is organized as follows: Section 2 analyses the problems concerning the development of e-government websites. Section 3 describes content management systems, the tools generally employed for creating, managing and updating an e-government website. Section 4 presents the theme of end-user development, the experiences of EUD in various application domains, and the characteristics of end users required to perform EUD activities. Section 5 describes our case study. Section 6 provides a further analysis related with the case study, which appears promising for future implementations of EUD-based techniques in the field of e-government websites. Section 7 briefly discusses the novel ideas proposed in the paper and Section 8 concludes the paper.

2 Characteristics of E-Government Websites

In many countries all over the world, several laws establish precise accessibility goals an e-government website must satisfy [39]. For instance, since 2005, websites of Italian government agencies must satisfy the requirements established in the "Disposizioni per favorire l'accesso dei soggetti disabili agli strumenti informatici" ("Provisions to support the access to information technologies for the disabled", Law no. 4 January 2004) [40].

Most of these laws on accessibility (including the Italian one) are based on the Web Content Accessibility (WCAG) release 1.0 recommendation of World Wide Web Consortium (W3C) [54]. WCAG 1.0 includes fourteen guidelines that must be followed by web developers to make their sites accessible to people with disabilities. Such people are (i) those who need to browse web pages by using assistive technologies because of physical disabilities (i.e. people having vision, hearing or mobility

impairments); (ii) elder people, who experience changes in vision, hearing, dexterity, and memory as they age; (iii) people who navigate the web through obsolete or limited hardware/software technologies, including old browser versions, low bandwidth connections to the Internet, mobile phones, personal digital assistants [53]. Most of the accessibility guidelines can be followed by writing proper HTML code, while others impose constraints to the interaction experience (for example by suggesting to avoid scripting code and to limit the presence of moving objects) or suggest the creation of simpler pages in terms of layout, graphics and language.

Furthermore, nowadays, e-government websites are becoming crucial also for the services they provide to citizens. Online services can be very different one another and dedicated to different kinds of users. For examples, some citizens may find easier to pay local taxes on the web, or may find useful reserving front office services or asking for personal documents by filling in forms on the website. Online services represent a particular kind of content that, like the other information to be published on the website, needs some domain knowledge generally possessed by personnel of the various agency departments.

The necessity to publish a huge and diverse amount of content usually requires a decentralized activity. To support distributed content authoring and easy website management, content management systems are generally adopted. Their main characteristics are discussed in the following section, along with their limitations in satisfying accessibility requirements and supporting end users in the creation of complex content.

3 Content Management Systems and Their Role in E-Government Website Creation and Maintenance

Like other kinds of web authoring tools, content management systems are becoming popular in relieving web site developers from low-level details of page design and implementation. In particular, content management systems implement the so-called *content-driven* paradigm: they support a separation between page presentation and page content, and allow users to ignore those aspects related with markup and scripting languages necessary to build the pages. As far as page presentation, pre-formatted templates are usually available in a CMS, as well as WYSIWIG functionalities to support users in creating new templates. But, more importantly for website content evolution, CMSs usually include tools that allow people not expert in information technologies to create web contents. Interacting with these tools is usually similar to interacting with office applications users are accustomed to use in their daily work. For example, a typical interaction could consist in choosing a page template and filling in it by editing objects or importing them from other sources.

These characteristics permit to create websites that satisfy usability requirements (such as consistency among pages, user-error management, system feedback), and which should be compliant with existing standards, particularly those related with page accessibility. Satisfying accessibility requirements established by national and international organizations is however a difficult task. Today several CMSs claim to directly support the development of websites compliant with WCAG guidelines. They are both open source tools, such as Plone [42], Moodle [34], Joomla! [24], Drupal

[15], TYPO3 [52], and commercial tools, such as Blackboard Content System [3], QnECMS [44], Sitekit CMS [49], Microsoft Content Management Server [32] or the more up-to-date Microsoft Office SharePoint [33] (an exhaustive survey of CMS products and technology is beyond the scope of the paper). In spite of these claims, for most of these tools, evaluating which WCAG guidelines are taken into account and how is not so easy. For example, exploring Joomla! documentation, one discovers that several accessibility fixes are due in the future version of the tool. The same is true for Moodle. However, Joomla! developers have already declared that some WCAG requirements are outside of Joomla! development team, as they have to be addressed by template designers or content managers [25]. A similar claim is made by Plone's development team, who is aware that a number of WCAG checkpoints are subjective, and thus their interpretation may vary [43]. In TYPO3, an *extension* (plugin) is available for managing content accessibility, but related user manuals are unavailable at the date of writing this paper, and thus it is not possible, also in this case, to evaluate a priori the suitability of TYPO3 for creating e-government websites. A thorough evaluation is even more difficult for commercial CMSs. Generally, their vendors declare that they are WAI compliant without giving further details.

Another topic concerning CMSs is the availability of server-side or client-side languages that can be used to personalize the website created through a CMS. However, such languages almost always require advanced skills, not only in computer programming, but also in information architectures of web applications. Therefore, they are used generally by professional software developers to implement the functionalities required for the domain at hand. In e-government websites, these functionalities include online services for tax payment, document request, front-office reservation, registration to public services (e.g. schools). In order to implement these functionalities and make them available on the web, software developers must always perform requirements gathering and analysis by interviewing the personnel of the interested department. This centralized approach and the misunderstandings arising among computer scientists and people with different competencies are often the reasons for delays in service development and publication. Authoring tools suitable to this task could adequately support the personnel at various departments in creating online services. Actually, some CMSs are specialized for particular domains, such as media sharing or personal spaces. These CMSs offer tailoring techniques or component-based methodologies to create web pages with functionalities for photo or movie sharing, guestbook management, meteorological forecasting. In a similar way, domain-dependent functionalities such as those offered by e-government websites could be designed directly by publishers.

4 End-User Development: From Desktop to Web Applications

The main goal of End-User Development is to study and develop techniques and applications for "empowering users to develop and adapt systems themselves" [28]. The level of complexity of these techniques should be appropriate to the users' individual skills and situations, and possibly allowing them to easily move up from less complex to more complex EUD activities. To this end, a classification of EUD activities has been proposed in [8][9] and further elaborated in [28]. The authors called

parameterization or *customization* all activities that allow users to choose among alternative behaviors already available in the application, resulting for example in associating specific computation parameters with specific parts of the data or in applying different functionalities to the data. Then, they classify as *program creation* or *modification* the EUD activities carried out through programming by example, incremental programming, model-based development, extended annotation.

4.1 EUD Solutions in Different Application Domains

EUD techniques have been used for many years in commercial software, such as macro recording in word processors, formula composition in spreadsheets or filter definition in e-mail clients [28]. However, on the one hand, they are far to be used extensively by a large community of end users, and, on the other hand, there exists the potential for employing EUD techniques in many other application domains and with different levels of complexity.

Research projects have been funded to design EUD techniques that support householders in programming their home appliances (e.g. digital radios, televisions, telephones) [4][5], in order to possibly obtain intelligent environments [14]. In the AutoHAN project [5], the idea is to use physical infrared remote controls that can become the syntactic elements in a program and that can be composed by the user to represent sophisticated functions. Component-based approaches for EUD are proposed in the field of computer-supported collaborative work [36], by providing visual tailoring environments that allow users to easily create search tools, chat tools and shared-to-do lists [55]. Repenning and Ioannidou propose agent-based programming as a paradigm for EUD [45]. They demonstrate the feasibility of this approach by applying it to many different domains, from game applications, to simulation environments, to software for education. Myers et al. are developing natural programming languages and environments to permit people to program by expressing their ideas in the same way they think about them [37]. They performed feasibility studies in the domain of video games, after having examined how children use and structure language to solve problems, and in the domain of business programming, after having analyzed how adults describe database access scenarios. Different techniques for EUD have been implemented in the software shaping workshops, end-user environments supporting domain experts in medical diagnosis [9], mechanical engineering [10] and geological forecasting [7]; such techniques are based on annotation mechanisms and visual programming through direct manipulation.

The area of EUD also involves the creation, modification and adaptation of web applications. This activity may turn out to be even more difficult than the development of traditional desktop applications, since it requires to know different markup languages, programming languages (both client and server side), interaction techniques with databases. In [48], the typical hurdles in web development have been identified, such as the stateless nature of the HTTP protocol and the necessity of session management, handling cross-platform compatibility, establishing and managing database connections, input validation. To overcome these problems, a software system, called Click [46], has been developed, which allows users to generate HTML code by simply instantiating and positioning components for a page under construction. In [30], Macías and Paternò propose an approach to the customization of

web-based applications, which exploits intelligent mechanisms to infer customization rules from user changes. In this case, end-user web developers who need to deal with structure and presentation of web pages are facilitated by an automatic system that builds an end-user profile containing customization preferences and then uses it to regenerate web pages according to such preferences. The use of wikis for EUD is instead advocated by Anslow and Rielhe [1]: wikis are regarded as a platform to support end users not only in contributing content, but also in performing computational tasks. They applied this technique for the development of business queries in web information systems. The work of Ginige and colleagues [13][22][27] is in the field of web information systems too. However, they propose a different solution: the definition of a meta-model of web applications and a set of form-based tools that can be used by end users to customize and evolve their applications, thus making the software architecture completely transparent to them. The ideas of meta-modelling and form-based EUD techniques seem very promising also in the application domain considered in this paper. However, while the tools described in [27] require users to follow precise syntaxes to create executable code, we propose here an evolution of the technique towards a more natural and direct manipulation interaction.

4.2 End Users' Characteristics

One of the most important activities when an interactive system is designed and evaluated is the characterization of its end users, especially if such end users are required to perform EUD activities.

Cypher defines end users as people who use a computer application as part of their daily life or daily work, but not interested in computers per se [12]. They can be technicians, clerks, analysts and managers who are often required, due to new organizational, business and commercial technologies, to perform end-user computing, i.e. "to develop software applications in support of organizational tasks" [6].

Some researchers focus the attention on end users with a high professionalism, such as interior designers [18], medical doctors [9], mechanical engineers [10], geologists [7], biologists [26], urban planners [2]. This has motivated the definition of a particular class of end users, the so-called *domain experts* [8][28], that is experts in a specific domain, not necessarily experts in computer science, who use computer environments to perform their daily tasks by acting as designers and being creative [21]. According to the spectrum presented in [56], they are *software developers using domain-specific languages* to write programs in order to solve specific problems that they own.

Web applications are often developed by "sophisticated end users" [48]: they are causal webmasters who, though possessing limited competencies in web technologies, are characterized by a strong sensibility and a deep motivation in creating their own artifacts [47]. They are sophisticated in that they are experienced in web design even though they find difficulties in managing the typical complexities in web development [48]. End users of wikis (e.g. Wikipedia), media sharing systems (e.g. Flickr) and other Web 2.0 systems [38] are classified as *web contents developers* in [56]; they share with casual webmasters the high motivation. In particular, they are very motivated in contributing their contents and collaborating through the web, and they are willing to spend time for preparing web material and publishing it.

As far as the development of e-government websites is concerned, the end users of content management systems represent another kind of web contents developers. However, they are not so motivated to create web material, but often perceive such activity as an overhead with respect to their daily work. The characteristics of these users, which we consider crucial to design adequate EUD techniques, are discussed more in detail in the next section.

5 EUD in E-Government Website Content Creation: A Case Study

During our collaboration with a large Italian municipality we had the opportunity to know and analyze the needs for EUD in e-government website content creation.

This municipality adopted in 2003 a commercial CMS to support content creation by the employees of various departments. A significant personalization work was performed to adapt the CMS to the specific context and customer's requirements. Clearly, the adoption of a more recent CMS product would ensure an improvement in website management, content creation and accessibility satisfaction. However, the huge amount of content to be migrated and the necessity of performing further personalization work and personnel training have discouraged until now managers and developers to make this choice.

During first informal conversations with some publishers, we discovered that they found many difficulties in creating web contents, mainly for two reasons: 1) their own characteristics; 2) the lack of some important functionalities in the CMS. As to the first point, from conversations it emerged that publishers belong to an heterogeneous population, which includes experts in different domains, thus having different competencies, skills, and cultural background. Most of them do not hold a higher education degree and their ages range in a wide spectrum. Publishers seem often to be insufficiently motivated in doing content authoring, by perceiving such activity as alien to their daily work. Moreover, they complained that, while interacting with the CMS adopted in their agency, they were often charged with housekeeping activities. For example, the creation of some type of content required publishers to edit directly the generated HTML code, in order to satisfy accessibility requirements defined in WCAG 1.0. These activities are natural for the computer expert and manageable by casual webmasters, but they are perceived as intricate by publishers, who not rarely arrive at refusing to perform the assigned content authoring tasks. Furthermore, most publishers do not perform these tasks frequently, depending them on deadlines for tax payments or other bureaucratic issues; therefore, such users tend to forget many details of the procedure to be followed, especially when it requires some editing of HTML code.

These difficulties suggested us that an approach to CMS development aimed at integrating EUD techniques in the CMS itself could overcome different kinds of problems in e-government website creation, management and updating.

Therefore, we implemented a simple EUD technique to solve a specific problem encountered by publishers; then, a case study research [57][58] was carried out to examine in-depth the interaction with the original CMS and to evaluate how the EUD-based approach improved the situation. In the following, we describe the problem

considered and a possible EUD solution. Then, the main results of the case study research are briefly presented (see [19] for more details).

5.1 EUD for Accessible Content Creation

To demonstrate the usefulness of EUD in the considered field, we faced the problem of creating tabular content to be published on an e-government website. Tabular content must satisfy guideline 5 “Create tables that transform gracefully” of WCAG 1.0 [54]. The six checkpoints of the guideline must be followed to support disabled people (users with blindness or low vision), who access tabular information through assistive technologies, such as a screen reader or a Braille display. The ability to produce accessible tabular content is of course a basic feature one would expect from a CMS (though this is not always the case). However, the EUD approach here proposed has a broader scope since it is suitable to support publishers in other and more sophisticated tasks.

To create accessible tables with the original CMS, publishers must modify the HTML code generated by the CMS. In particular, the interaction occurs as follows. The authoring tool available in the CMS provides a button in a toolbar to activate table creation. When the user selects this button, the system presents the user with a dialog window that asks for inserting the number of rows and columns of the new table. After interacting with this dialog window, a “prototype” table is created showing cells whose content is “Col 1 Row 1”, “Col 2 Row 1”, and so on for the first row, “Col 2 Row 1”, “Col 2 Row 2”, and so on for the second row, for all the rows requested by the user. Figure 1 shows the table created when the user asks for a two rows-two columns table.

When the user clicks on a table cell, its content is selected and the user can substitute it with the desired content. For example, let us suppose that the publisher inserts person names (Maria, Paola) and surnames (Rossi, Bianchi) in the first column and

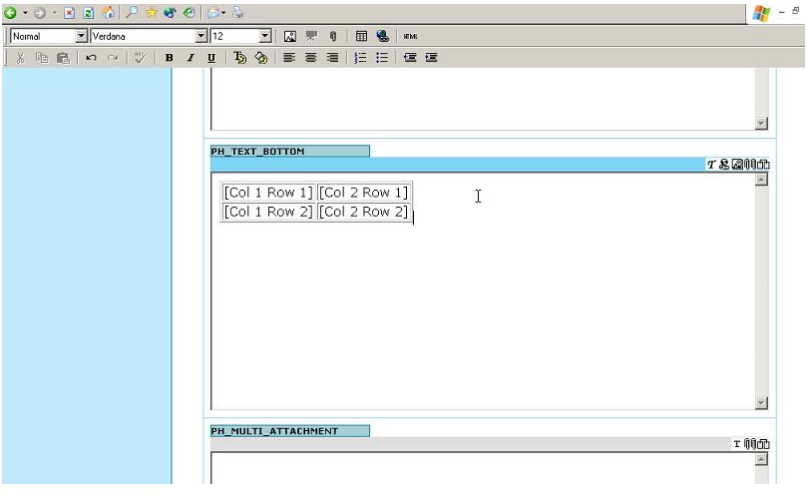

Fig. 1. The table created as a consequence of user request

second column respectively. The resulting HTML code underlying the table would be the following:

```
<TABLE>
     <TR>
            <TD> Maria </TD>
            <TD> Rossi </TD>
     </TR>
     <TR>
            <TD> Paola </TD>
            <TD> Bianchi </TD>
     </TR>
</TABLE>
```

To make this code compliant with guideline 5 of WCAG 1.0, the publisher must access this code through the proper button in the CMS toolbar and modify it as follows (users' modifications to the code generated by the CMS are highlighted):

```
<TABLE SUMMARY="This table contains name and surname of the
employees that work at the Public Relations Department of the
Brescia municipality">
        <CAPTION>Employees working at the Public Relations
      Department
        </CAPTION>
            <TR>
                  <TH ID="name">Name</TH>
                  <TH ID="surname">Surname</TH>
            </TR>
            <TR>
                  <TD headers="name">Maria</TD>
                  <TD headers="surname">Rossi</TD>
            </TR>
            <TR>
                  <TD headers="name">Paola</TD>
                  <TD headers="surname">Bianchi</TD>
            </TR>
</TABLE>
```

Actually this code aims at satisfying three of the six checkpoints of guideline 5 of WCAG and in particular:

- A tag <TH> has been added for each column by specifying the column header as a value of attribute ID of <TH>. This is to satisfy checkpoint 1 of WCAG guideline 5;
- Each cell identified by an element <TD> has been associated with the corresponding column by using attribute headers, whose value must correspond to the column header. This is to satisfy checkpoint 2 of WCAG guideline 5;
- Attribute summary has been added in tag <TABLE> and element <CAPTION> has been inserted as first child of element <TABLE>, in order to satisfy checkpoint 5 of WCAG guideline 5.

According to the municipality managers, these checkpoints are the minimum requirements to be satisfied by tables published in their website. In fact, checkpoint 6 gives additional indications for managing table linearization, but it has priority 3, while it was decided that, at the moment, the goal was to obtain Conformance Level AA for the website [53]. Checkpoints 3 and 4 refer to the use of tables for page layout; they are satisfied a priori, since page layout is managed through style sheets.

To support publishers in this work practice, they were provided with a paper-based manual, which presented a detailed example to be adapted to the case at hand. Notwithstanding this, this activity represented a problem for publishers.

In order to solve the problem by implementing a simple EUD technique based on parameterization, the source code of the CMS was properly modified. The interaction with the system for inserting a table now occurs as follows. When the user selects the tool to insert a table, after the dialog window asking for the number of rows and columns, a new dialog window is presented to the user. Such window, shown in Figure 2, asks the user for inserting a text for the caption, a text for the summary, and as much headers as the number of columns previously declared. The window autonomously adapts its size according to the number of column headers that need to be requested. It also helps the user to insert caption and summary by remembering her/him the meaning of such information directly in the text fields. The user is obliged to fill in all the text fields: in fact, when s/he selects the OK button, information are checked and, if one is missing, a warning message is presented to the user, by constraining her/him to return to the dialog window and complete data insertion. This relieves the publisher from checking the correctness of the table, making it easy to assess her/his EUD activity [28].

The new procedure is clearer and easier for people not expert in computer technologies. This was confirmed by the results of our case study research.

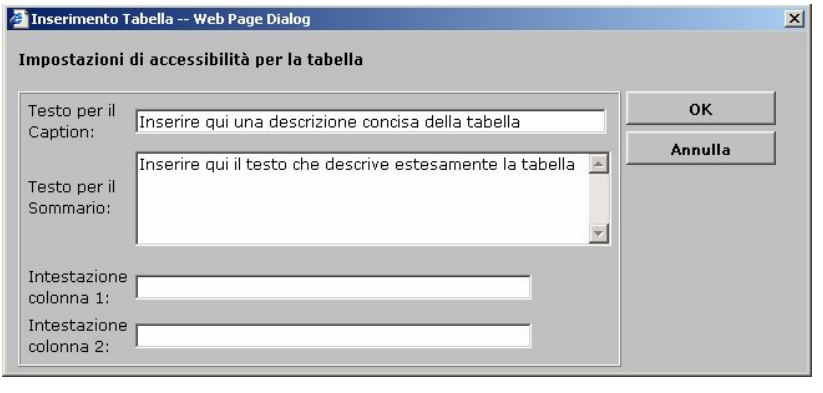

Fig. 2. The dialog window asking for accessibility parameters

5.2 Case Study Research: Methodology and Results

The goal of the case study research was to investigate the difficulties encountered by publishers in creating accessible content and to evaluate the benefits of enhancing the adopted CMS with EUD features. Therefore, our research questions aimed at

understanding what happened during content creation and how the publishers faced the problems of the CMS about content accessibility. Then, our goal was also investigating if the extension of the CMS with the EUD feature would have provided significant improvements in terms of both performance and publishers' willingness to carry out activities concerned with accessible table creation.

We involved eight users chosen from different departments of the municipality. The sample, even if little, can be considered enough representative of the publisher population, because it included people expert in different domains and having different competencies, skills, and cultural background. Participants were asked for performing a task concerning the creation of an accessible table. A within groups technique was adopted, meaning that all users in the sample performed the same task using the original CMS and the extended CMS. To avoid polarization due to learning effects, different execution orders were defined.

Data gathering was carried out in the usual work place of users. The techniques adopted were *structured interviews* (after the execution of the assigned task in the two sub-cases), *observation* during task execution and *performance measures* (completion time and number of errors). The first two techniques were meant to provide us with a qualitative evaluation, while the last were meant to provide us with quantitative data possibly corroborating qualitative ones. An evaluation form was prepared to gather both qualitative and quantitative data: it included few simple questions (e.g. "Which are the main difficulties you encountered using the original CMS?", "Which are the main improvements you noticed in the new solution?") and some fields to be filled in by the observer with the observations taken during task execution and with the quantitative data.

During the execution of the task with the original CMS, we observed that most of publishers applied mechanically what was suggested by the example table in the manual and made several mistakes during the adaptation of that table. They often forgot to change some parts of the table, by leaving the information already present in the example table. Moreover, by listening to users' spontaneous comments, we discovered that difficulties and misunderstandings arose because users did not understand the meaning of tags correctly. For example, they confounded the terms `caption`, `headers` and `summary` (maybe this was also due to the translation between English and Italian). The interviews confirmed that publishers perceived the task as too difficult and requiring an exaggerate effort. In general, publishers considered the HTML code manipulation as a work alien to their competencies and tasks, useless, and time consuming. Most of them explicitly declared that they were not willing to spend energies and time in doing that job.

The new approach apparently solves all the above problems. Participants commented that the system now requires them exactly what is needed, it does not ask them anymore for thinking about the accessibility parameters, but it just drives them through table creation, and thus it also avoids them to consult the manual. These positive results, gathered through direct observation and interviews, are corroborated by quantitative data: the comparison of completion times and error numbers demonstrate that, using the procedure offered by the extended CMS, there is a significant improvement in both robustness and efficiency [19].

6 Toward Online Service Creation through EUD

We carried on the collaboration with the municipality with the aim of finding a solution also to the problem of online service development. This task is still at the hands of personnel of the Computer Science department of the municipality, since the CMS does not provide proper facilities for transferring it to publishers. This should remain valid also if a more up-to-date CMS would be adopted, since, as already mentioned, existing products provide scripting languages and macro languages that only computer experts or power users are able to manage.

This section describes a possible EUD approach to developing online services. The approach stems from the analysis of how personnel of the Computer Science department operate to create such services and from the characterization of publishers carried out during the case study research. We used unstructured interviews with a representative software developer to elicit knowledge about the kinds of services to be made available on a municipality website and about their design and development.

From the analysis of online services currently offered to citizens on the website it is possible to obtain the following classification: 1) front office reservation; 2) tax payment; 3) document request; 4) document submission; 5) registration to courses or schools. All these services are accessed by the end users of the website through form-based pages, since fill-in form interaction style has a low cognitive burden for most of people and it is easy to implement. More precisely, end users are presented with the forms composing a service through a *step-by-step instruction design pattern* [51]. This permits to drive users through the task, in order to acquire all necessary information in each step, and to perform validity checks on input data. Also inspired by the work of Ginige et al. [13][22], we think that an interaction style based on fill-in forms and a step-by-step instruction design pattern could be also at the basis of the EUD technique allowing publishers to create online services.

We illustrate this technique by an example. Currently, front office reservation services are implemented in the municipality website as a 5-step wizard, where the steps are: 1) counter choice; 2) date choice; 3) time choice; 4) input of personal data; 5) summary of data. The first three steps are implemented through radio buttons permitting exclusive choices; the fourth step presents text fields and combo boxes to input data; the fifth just presents all inserted data and asks for a confirmation. In each step, but the first, it is possible to go back one step to modify previous inserted data. An area on the right side of the page shows the steps performed, the step currently under compilation, and the steps remaining. Figure 3 shows the website page during the reservation of general registry office services: step 3 (time choice) is under compilation (see the main area of the page); on the right side, the white box and the symbol highlight the step under compilation, whilst previous steps (counter selection and date selection) are marked as done ( ) and next steps (input of personal data and summary of data) are marked as to be done ( ).

We argue that the implementation of such kind of wizards could be performed easily by publishers if the CMS supports them through a step-by-step form-based interaction. For creating a new online service, the publisher will first choose the class of the service, for example the “front office reservation” class. Then, the system will drive him/her through the steps for creating the service by means of fill-in forms. The

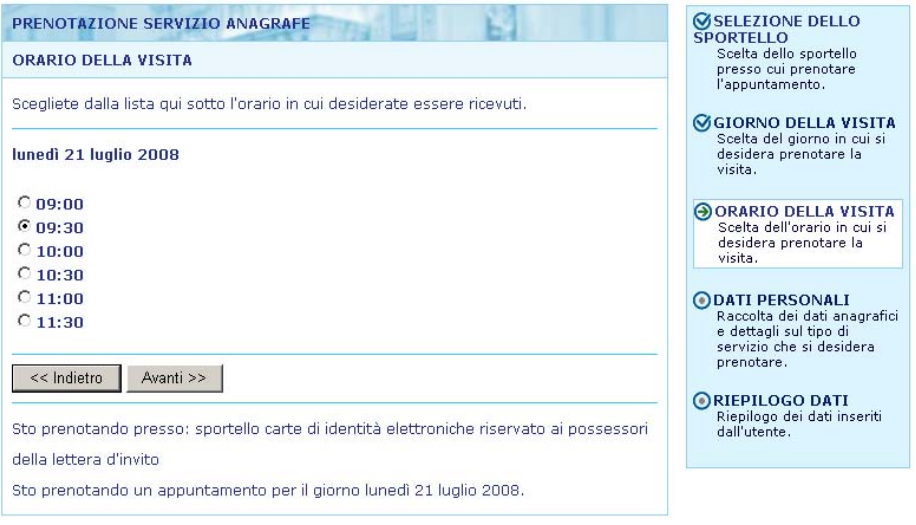

Fig. 3. Step 3 of front office reservation

first step will consist in generating the list of counter choices that pertain to the publisher's department. To this aim, a list of all counters could be available and the publisher might move items from this list to a list of selected counters, which will be presented on the website as a set of radio buttons. Figure 4 shows a mock-up of this EUD solution. (For the sake of paper readability, all mock-ups are in English).

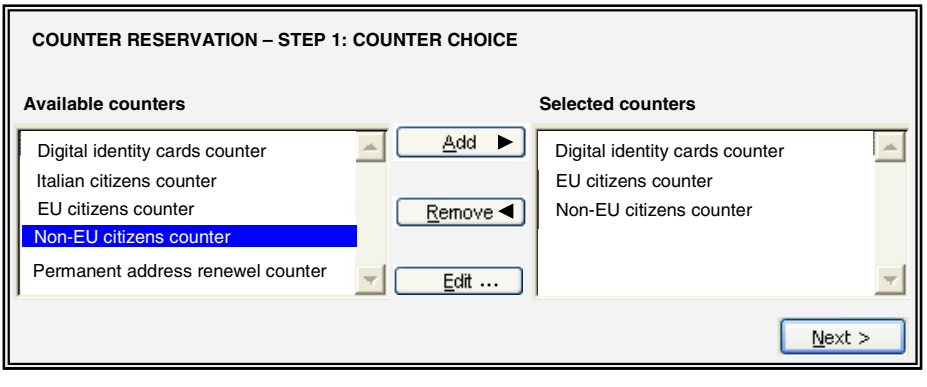

Fig. 4. Mock-up of the EUD tool for generating a list of radio buttons related with counter choice

The next step should be a calendar component customized to publishers' needs: it should support the choice of start and end dates, the indication of holidays, and the choice of week days in which the counter is open. Figure 5 shows a mock-up of this EUD solution: three calendars are used to choose start, end, and holidays; six check boxes permit to choose working days. The system must generate a set of radio buttons for date choice that satisfy all constraints defined by the publisher.

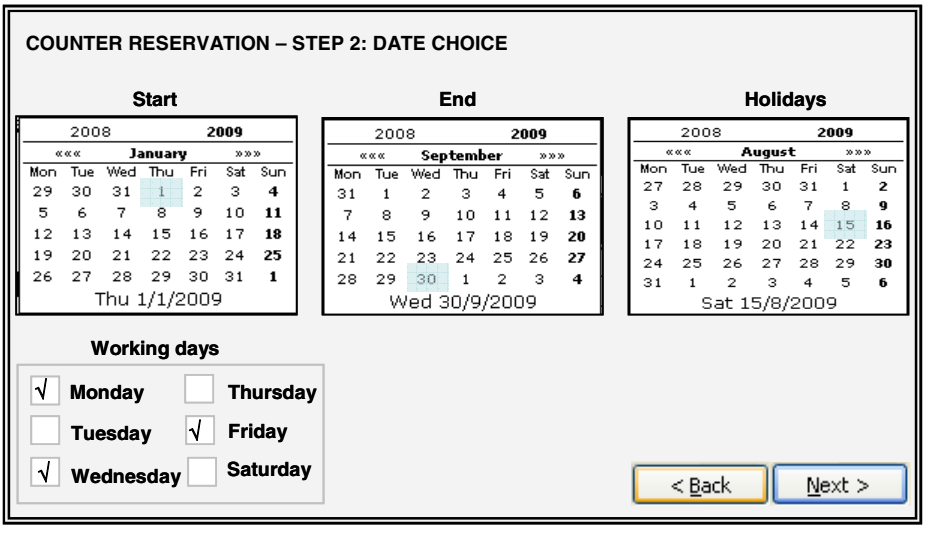

Fig. 5. Mock-up of the EUD tool for generating a list of radio buttons related with date choice

The third step should be the creation of time choices: the publisher could choose start, end, break and time intervals, and the system should generate the list of all possible times as a set of radio buttons satisfying the established constraints. Figure 6 shows a mock-up of this EUD solution: combo boxes permit time choices.

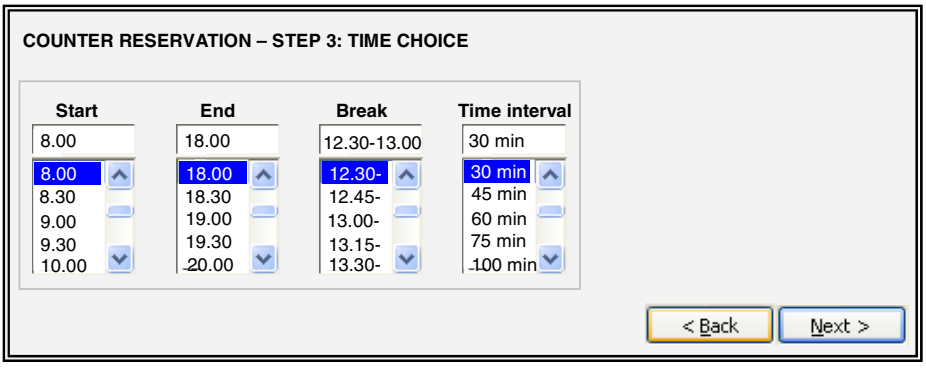

Fig. 6. Mock-up of the EUD tool for generating a list of radio buttons related with time choice

Then the publisher should create the form for input data; also in this case, the solution foreseen for counter choice, i.e. the double-list control, can be useful: a list of all possible personal data could be available, giving the publisher the possibility to select only those s/he considers necessary to be requested in the case at hand. Finally, the step summarizing the inserted data could be generated automatically by the system on the basis of the previous steps created by the publisher.

7 Discussion

The design of EUD techniques proposed in the previous sections derives from a basic observation: end users who has to carry out some software development in the domain of e-government websites are different from casual webmasters or power users. Therefore, they must not be aware of performing software development. They should accomplish tasks that consist in just editing content (including providing information that become HTML attribute values) or selecting some content from available choices. The underlying system will be in charge of generating the correct code by exploiting the content provided by the user.

Both techniques presented in Sections 5 and 6 are based on fill-in form interaction style, which has been judged suitable for the considered end users, namely personnel used to carry out administration tasks often consisting in the compilation of paper-based forms. Therefore, this interaction style should be natural and simple for these end users, by allowing them to operate according to their mental models of the activities to be performed [11]. On the other hand, this EUD style has been applied also in other contexts with successful results [27], even though, differently from [27], we aim to drive users through all their choices, without asking them to remember some particular syntax for providing the information requested by the system.

In the case of online service creation, the fill-in form interaction style is combined with a step-by-step instruction design pattern that reflects the structure of the service to be created. The publisher should not have so much freedom (and consequent responsibility) to modify the layout of the service pages or the structure of the service. Moreover, s/he should not access the generated code.

The idea of an EUD technique based on a step-by-step instruction design pattern actually arises from the analysis of the output to be generated (i.e. the service to be made available to website's users). This analysis produces a model of the service, which helps to determine the most natural way for publishers to take care of its development.

Service analysis and model-based design of EUD techniques should remain at the hands of the software developers belonging to the organization managing the website: they should identify the classes of services that could be developed, the steps constituting each class, and the elements composing each step. Then, their work will be the development of the fill-in forms that allow publishers to create online services. These activities can be characterized as *meta-design* activities [17], in that they are carried out to "design the design process" [21], i.e. to design the EUD activities that publishers should perform. In [17], this is regarded as the first level of meta-design, which refers to the possibility offered to end users to transform and modify components and contents at use time, according to emerging needs and tasks, as it may happen in the case of online service creation. Methodologically, software developers operate at a meta-level by "establishing the conditions that will allow users, in turn, to become designers" [17].

Obviously, software developers may become again a bottleneck in those contexts that are very dynamic, where classes of services evolve over time and new classes must be designed and developed frequently. However, the approach here presented seems – at least at the moment – suitable to the dynamicity of the considered application domain. Its scalability should be studied if one would like to extend its application to

other domains, such as e-commerce or e-learning websites. Further research on meta-design approaches could help to find solutions to this problem.

8 Conclusions

This paper focused on a particular application domain, namely the creation and maintenance of e-government websites. In this domain, distributed content authoring is often a necessity to avoid delays in publishing important information for citizens and to overcome communication gaps between software developers and domain experts.

However, despite the use of CMSs, this distributed activity is far to be performed in an easy, efficient and effective way. By analyzing the case of an Italian municipality we discovered that publishers find several difficulties in creating accessible contents. Therefore, a CMS extension with a simple EUD technique has been developed to eliminate such difficulties. The approach has been evaluated with publishers, by giving positive results [19].

We also observed that the creation of more sophisticated contents, e.g. online services, are still at the hands of software developers, since this task appears as too difficult to be carried out by publishers. In this paper, we tried to demonstrate the contrary: it is possible to implement proper EUD techniques devoted to these end users, whose motivation and interest in software development is low. From our analysis, it emerged that the fill-in form interaction style and the step-by-step instruction design pattern could be adopted to design EUD techniques for the domain at hand. They have been proved successful in the case of HTML editing for accessibility satisfaction, and they seem to be promising in the case of online service development.

As future work we plan to implement and test the mock-up ideas presented in this paper, as well as to identify and define meta-design guidelines for the design of EUD techniques by personnel of the Computer Science municipality department.

Acknowledgments. The author wishes to thank Sergio Colosio of Comune di Brescia, Italy, and Loredana Parasiliti Provenza of Università di Milano, Italy, for the fruitful discussions about the content of this paper. She is also indebted to Matteo Sacco for the development of the CMS extensions and to the publishers of the Comune di Brescia for their availability in participating in interviews and tests of the implemented EUD technique.

References

1. Anslow, C., Rielhe, D.: Towards End-User Programming with Wikis. In: Proc. WEUSE IV 2008, Leipzig, Germany, pp. 61–65 (2008)
2. Arias, E., Eden, H., Fischer, G., Gorman, A., Scharff, E.: Transcending the Individual Human Mind - Creating Shared Understanding through Collaborative Design. ACM Transactions on Computer-Human Interaction 7(1), 84–113 (2000)
3. Blackboard Content System, `http://www.blackboard.com/products/Academic_Suite/Content_System/index`

4. Blackwell, A.F.: End-User Developers at Home. Communications of the ACM 47(9), 65–66 (2004)
5. Blackweell, A.F., Hague, R.: AutoHAN: An architecture for programming at home. In: Proc. IEEE Symposium on Human-Centric Computing Languages and Environments, pp. 150–157 (2001)
6. Brancheau, J.C., Brown, C.V.: The Management of End-User Computing: Status and Directions. ACM Computing Surveys 25(4), 437–482 (1993)
7. Carrara, P., Fogli, D., Fresta, G., Mussio, P.: Toward overcoming culture, skill and situation hurdles in human-computer interaction. Int. J. Universal Access in the Information Society 1(4), 288–304 (2002)
8. Costabile, M.F., Fogli, D., Letondal, C., Mussio, P., Piccinno, A.: Domain-Expert Users and their Needs of Software Development. In: Proc. UAHCI Conference, Crete, pp. 232–236 (2003)
9. Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A.: End-User Development: the Software Shaping Workshop Approach. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End-User Development, pp. 183–205. Kluwer Academic Publisher, Dordrecht (2006)
10. Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A.: Visual Interactive Systems for End-User Development: a Model-based Design Methodology. IEEE Transactions on Systems Man and Cybernetics, part A- Systems and Humans 37(6), 1029–1046 (2007)
11. Costabile, M.F., Mussio, P., Parasiliti Provenza, L., Piccinno, A.: End Users as Unwitting Software Developers. In: Proc. WEUSE IV 2008, Leipzig, Germany, pp. 6–10 (2008)
12. Cypher, A.: Watch What I Do: Programming by Demonstration. MIT Press, Cambridge (1993)
13. Da Silva, B., Ginige, A.: Modeling Web Information Systems for Co-Evolution. In: Proc. ICSOFT 2007, Barcelona, Spain (2007)
14. De Ruyter, B., Van de Sluis, R.: Challenges for End-User Development in Intelligent Environments. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End-User Development, pp. 243–250. Kluwer Academic Publishers, Dordrecht (2006)
15. Drupal, `http://drupal.org/`
16. EUD-Net Thematic Network, `http://giove.cnuce.cnr.it/eud-net.htm`
17. Fischer, G., Giaccardi, E.: Meta-Design: A Framework for the Future of End User Development. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development, pp. 427–457. Kluwer Academic Publisher, Dordrecht (2006)
18. Fischer, G.: Seeding, Evolutionary Growth and Reseeding: Constructing, Capturing and Evolving Knowledge in Domain-Oriented Design Environments. Int. J. Automated Software Engineering 5(4), 447–464 (1998)
19. Fogli, D., Colosio, S., Sacco, M.: Managing Accessibility in Local E-government Websites through End-User Development: A Case Study. Int. J. Universal Access in the Information Society (to appear)
20. Folmer, E., van Welie, M., Bosch, J.: Bridging patterns: An approach to bridge gaps between SE and HCI. J. of Information and Software Technology 48(2), 69–89 (2005)
21. Giaccardi, E., Fischer, G.: Creativity and Evolution: A Metadesign Perspective. Digital Creativity 19(1), 19–32 (2008)
22. Ginige, A., De Silva, B.: CBEADS©: A Framework to Support Meta-design Paradigm. In: Stephanidis, C. (ed.) HCI 2007. LNCS, vol. 4554, pp. 107–116. Springer, Heidelberg (2007)
23. Goette, T., Collier, C., Daniels White, J.: An exploratory study of the accessibility of state government Web sites. Int. J. Universal Access in the Information Society 5, 41–50 (2006)
24. Joomla!™, `http://www.joomla.org/`

25. Joomla! Help Site – WCAG Checklist, http://help.joomla.org/
26. Letondal, C.: Participatory Programming: Developing Programmable Bioinformatics Tools for End-Users. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End-User Development, pp. 207–242. Kluwer Academic Publishers, Dordrecht (2006)
27. Liang, X., Ginige, A.: Enabling an End-User Drive Approach for Managing Evolving User Interfaces in Business Web Applications. In: ICSOFT 2007, Barcelona, Spain (2007)
28. Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-User Development: An Emerging Paradigm. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End-User Development, pp. 1–8. Kluwer Academic Publishers, Dordrecht (2006)
29. Lieberman, H., Paternò, F., Wulf, V. (eds.): End-User Development. Kluwer Academic Publishers, Dordrecht (2006)
30. Macías, J.A., Paternò, F.: Customization of Web applications through an intelligent environment exploiting logical interface descriptions. Interacting with Computers 20, 29–47 (2008)
31. Majhew, D.J.: Principles and Guideline in Software User Interface Design. Prentice-Hall, Englewood Cliffs (1992)
32. Microsoft Content Management Server, http://www.microsoft.com/cmserver/default.mspx
33. Microsoft Office SharePoint Designer 2007 (2007), http://office.microsoft.com/it-it/sharepointdesigner/FX100487631040.aspx
34. Moodle, http://moodle.org/
35. Mørch, A.: Three Levels of End-User Tailoring: Customization, Integration, and Extension. In: Kyng, M., Mathiassen, L. (eds.) Computers and Design in Context, pp. 51–76. MIT Press, Cambridge (1997)
36. Mørch, A., Stevens, G., Won, M., Klann, M., Dittrich, Y., Wulf, G.: Component-Based Technologies for End-User Development. Communications of the ACM 47(9), 59–62 (2004)
37. Myers, B.A., Pane, J.F., Ko, A.: Natural Programming Languages and Environments. Communications of the ACM 47(9), 47–52 (2004)
38. O'Really: What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software, http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html
39. Paris, M.: Website accessibility: a survey of local e-government websites and legislation in Northern Ireland. Int. J. Universal Access in the Information Society 4, 292–299 (2006)
40. Parlamento Italiano, Disposizioni per favorire l'accesso dei soggetti disabili agli strumenti informatici, Legge 9 gennaio, n. 4, G.U. n. 13 del 17 gennaio (in Italian) (2004) (in English), http://www.pubbliaccesso.it/normative/law_20040109_n4.htm
41. Petre, M., Blackwell, A.F.: Children as Unwitting End-User Programmers. In: Proc. VL/HCC 2007, Coeur d'Alène, USA, pp. 239–242 (2007)
42. Plone™, http://plone.org/
43. Plone™– Accessibility Statement, http://plone.org/accessibility-info
44. QnECMS – Quick & Easy Accessible CMS, http://www.qnecms.co.uk/
45. Repenning, A., Ioannidu, A.: Agent-Based End-User Development. Communications of the ACM 47(9), 43–46 (2004)
46. Rode, J., Bhardwaj, Y., Pérez-Quinones, M.A., Rosson, M.B., Howarth, J.: As Easy as "Click": End-User Web Engineering. In: Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, pp. 478–488. Springer, Heidelberg (2005)

47. Rode, J., Rosson, M.B., Pérez Quinõnes, M.A.: End User Development of Web Applications. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End-User Development, pp. 161–182. Kluwer Academic Publishers, Dordrecht (2006)
48. Rosson, M.B., Ballin, J., Nash, H.: Everyday Programming: Challenges and Opportunities for Informal Web Development. In: Proc. VL/HCC 2004, Rome, Italy, pp. 123–130 (2004)
49. Sitekit CMS, http://www.sitekit.net/
50. Sutcliffe, A., Mehandjiev, N. (Guest eds.): End-User Development. Communications of the ACM 47(9), 31–32 (2004)
51. Tidwell, J.: Common Grounds: A Pattern Language for Human-Computer Interface Design, http://www.mit.edu/~jtidwell/common_ground.html
52. Typo3, http://typo3.com/
53. Web Accessibility Initiative, http://www.w3.org/WAI/
54. Web Content Accessibility Guidelines 1.0, W3C Recommendation (May 5, 1999), http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505
55. Won, M., Stiemerling, O., Wulf, V.: Component-Based Approaches to Tailorable Systems. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End-User Development, pp. 115–141. Kluwer Academic Publishers, Dordrecht (2006)
56. Ye, Y., Fischer, G.: Designing for Participation in Socio-Technical Software Systems. In: Stephanidis, C. (ed.) HCI 2007. LNCS, vol. 4554, pp. 312–321. Springer, Heidelberg (2007)
57. Yin, R.K.: Case study research: Design and methods. Sage, Newbury Park (1984)
58. Yin, R.K.: Case study methods. In: Green, J.L., Camilli, G., Elmore, P.B. (eds.) Handbook of complementary methods in education research, pp. 111–122. Lawrence Erlbaum Associates, Hillsdale (2006)