XIDE: EXPANDING END-USER WEB DEVELOPMENT

Evgenia Litvinova, Markku Laine and Petri Vuorimaa
Department of Media Technology, Aalto University, P.O. Box 15400, FI-00076 Aalto, Finland

Keywords: End-user Programming, Web Development, Unified Programming Model, Markup Languages, XForms.

Abstract: Most Web end-user development tools assume that users create Web applications entirely visually. However, such tools have a limited functionality and support inadequately the transition from visual editing to source code editing. In this paper, we introduce a tool, XIDE, that supports end users in creation of database-driven Web applications. The proposed tool is based on three main approaches: (1) visual composition of applications from reusable components, (2) source code editing of both applications and components, and (3) unified programming model based on markup languages.

1 INTRODUCTION

The increasing popularity of the Internet and Web 2.0 has made end users more familiar with the concepts and possibilities of the Web. End users occasionally use markup languages for writing in forums and wikis, creating mashups, editing media rich blog pages, and creating basic HTML pages. Besides static content, users want to create interactive database-driven applications to assist them in their daily lives (Rosson et al., 2005; Cypher et al., 2009).

Developing such applications from scratch is complicated for end users, mainly because it involves using of several programming languages and combining them together (Cardone et al., 2005). Visual tools, such as survey builders, mashup editors or component-based tools, assist end users (Ko et al., 2011). End users can create applications using wizards and visual manipulations without making any manual changes in the source code (Grammel and Storey, 2008). However, if the demanded functionality of the application cannot be achieved using a visual editor, a user is forced to edit the background source code. This task immediately requires advanced knowledge of a variety of technologies and concepts (Zang et al., 2008).

In this paper, we focus on active Web users, who already have an experience in using end-user development tools. We aim to expand the scope of applications they are capable of creating without them facing major learning challenges.

We address this problem with an end-user development tool, which combines the following approaches. First, it is a component-based tool with a visual editor. Second, unlike other visual tools, it does not aim to hide source code editing. Instead, it facilitates end users to do changes in the source code. Third, a background technology used in the tool has lower complexity and allows users to utilize the concepts they are already familiar with.

As a proof-of-concept implementation, we developed a prototype tool called XIDE, which uses a unified markup-based platform called XFormsDB as a background technology. Since XFormsDB is based on HTML[1], end users can leverage their knowledge of markup language concepts. We validate this approach by conducting preliminary user studies.

Our contribution is twofold. Firstly, we present a classification of end-user Web development activities. Secondly, our main practical contribution is the component-based architecture founded on a unified background technology, which enables end-user code editing and component modification.

The paper is organized as follows. The next Section discusses different levels of end-user Web development activities. Requirements for a background technology and the tool are presented in Section 3. A proof-of-concept implementation is described in Section 4 and evaluated in Section 5. Section 5 also contains the discussion and directions for future work. Finally, Section 6 presents conclusions.

[1]XFormsDB extends XForms 1.1, which is part of XHTML 2.0

Litvinova E., Laine M. and Vuorimaa P..
XIDE: EXPANDING END-USER WEB DEVELOPMENT.
DOI: 10.5220/0003934201230128
In *Proceedings of the 8th International Conference on Web Information Systems and Technologies* (WEBIST-2012), pages 123-128
ISBN: 978-989-8565-08-2

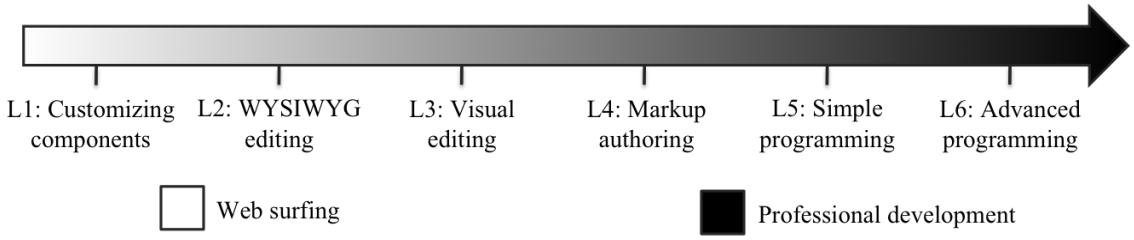

Figure 1: Spectrum of end-user Web development activities.

2 LEVELS OF END-USER WEB DEVELOPMENT ACTIVITIES

Traditionally, end users and developers were considered as two mutually exclusive groups. In the Web, these are *pure end users*, who do *Web surfing*, and *educated expert developers*, who do *professional development* and implement software products for other people. However, nowadays end users are more and more involved in actual Web development activities in order to solve their problems by themselves, and thus become end-user programmers (Ko et al., 2011).

We expanded the idea of end-user development activities classification described in (Costabile et al., 2008) by applying it to the Web. We define six levels of end-user Web development activities between Web surfing and professional development, as depicted in Figure 1. Each level defines activities a user is able to perform and corresponding skills a user needs to possess.

L1: Customizing Components. Users adjust existing Web applications by setting values to parameters, e.g., adjusting colors on a personal blog page.

L2: WYSIWYG Editing. Users develop static Web pages or targeted applications, e.g., surveys, in What You See Is What You Get (WYSIWYG) editors. They design only the visual representation of resulting applications using drag-n-drop and applying styles. Example tools are Google Sites[2], a page creation tool, and JotForm[3], a form builder.

L3: Visual Editing. Users create applications visually by adding, customizing, and connecting components, to compose a final application (Repenning and Ioannidou, 2006). Common examples are mashup editors, such as Yahoo! Pipes[4] and DashMash (Cappiello et al., 2011), and component-based editors (Won et al., 2006). The functionality of visual tools is strongly limited to the library of components. The visual editing does not require programming skills or syntax knowledge, however users often need to understand the programming paradigm of a tool and the logic of how applications are created.

L4: Markup Authoring. Users contribute to wiki and write advanced posts to blogs using plain markup languages. This is a code editing activity, however, used markup languages are often simplified for better understanding.

L5: Simple Programming. Users, who are familiar with programming languages, extend the limitations of visual tools. They do not have enough skills to develop applications from scratch, but they can modify the background source code of visually generated applications or predefined components.

In Dreamweaver[5], users can create a page visually in a drag-n-drop editor and manually edit its auto-generated source code to make more advanced changes. However, end users complain about difficulties while switching between visual to source code editors. (Park and Wiedenbeck, 2010).

In WordPress[6], users can create interactive applications with wide functionality using predefined templates and widgets. The background technology of the widgets is PHP, which is rarely known among users with skill level 5. Users need to know WordPress internal architecture and API before they can do even minor changes to a widget.

In Click (Rode et al., 2005), users can create data collection and management Web applications using component-based approach. The tool provides several layers of modification complexity, starting from customizing templates to modifying and extending the component framework by editing PHP code.

L6: Advanced Programming. Users are amateur programmers, but they know several complementary technologies and are able to construct interactive client-server Web applications for their personal use. Users are often forced to use professional tools and

[2]Google Sites, sites.google.com
[3]JotForm, www.jotform.com
[4]Yahoo!Pipes, pipes.yahoo.com/pipes
[5]Dreamweaver, adobe.com/products/dreamweaver.html
[6]WordPress, wordpress.org

Integrated Development Environments (IDEs). End-user tools for creating complex Web applications, including server-side logic, and database, are missing.

3 REQUIREMENTS

In this paper, we address the problem of smoothing the transition from the level 4 to the level 5 and 6. We aim to allow level 4 users to create Web applications, which are more advanced than those they could afford previously - including server-side and database functionality. It is a challenge because end users natively have an opportunistic approach to their tasks (Mosemann and Wiedenbeck, 2001). They attempt to approach the task with their current skills, and often lack a motivation to learn new technologies. However, when entering the level 5, users face problems, such as the limitations of visual tools, new background technology language, and missing server-side functionality.

Based on the review of end-user activities presented in Section 2, we propose a tool with a component-based architecture. In such a tool, users can create applications with higher functionality that they can attain by doing the development from scratch. Users can modify existing components to adjust them to the task. We aim to design a tool that supports editing activities from previous levels (1–3), such as parameter editing, visual editing, WYSIWYG editing, etc. Most Web development tools assume that end users avoid source code editing (levels 4 and 5). Because of this, the background technology and architecture are often selected without any consideration of end-user needs. We especially focus on source code editing activities.

R1: Background Technology must Be Unified and based on Markup Languages. The need of learning and combining different technologies is a problem for end users. Using fewer technologies or even one programming language could ease the development and client-server communication (Laine et al., 2011).

If the background technology looks familiar and allows to leverage existing knowledge, end users can start understanding and editing the source code without preliminary training. In this paper, we focus on the level 4 users, i.e. those who have some experience in markup editing. Thus, markup languages, such as HTML and XML, qualify as a basic technology for the end-user Web development. However, currently pure HTML supports only the creation of static pages only.

R2: Applications and Components both must Use the Same Background Technology. Source code editing activities include both applications and components code editing. If the background technology in components is the same as in applications, there is no additional learning barrier for end users when they decide to edit components. Moreover, components can be considered not only as bricks to build applications from, but also as examples of what can be achieved with the technology (Park and Wiedenbeck, 2010) .

R3: The Architecture of both Components and Applications must Be Similar and Transparent. When users edit components or contribute new ones, no additional knowledge must required to understand how to use a component and how to edit it.

R4: A Transition from Visual Editing to Source Code Editing must Be Smooth. Considering how end users approach the development process, a tool that is used on levels 5 and 6 must be extended with several important features, such as code editing activities assistance, immediate feedback on user actions, full-functional preview, highlighting of the link between the source code and the result.

4 PROOF-OF-CONCEPT IMPLEMENTATION

Based on the requirements, we designed and implemented a tool called XIDE. In this section, we present the tool and describe the background technology it uses for both applications and components.

4.1 XIDE

XIDE[7] is a tool for end-user Web development that assists end users in developing advanced, interactive database-driven Web applications. XIDE supports all development activities displayed in Figure 1 and fulfills the requirement R4. It aims to gently lead end users from the level 4 to the level 5 and 6 without them facing any major learning barriers.

In order to achieve this goal, XIDE combines three approaches. First, it helps end users to leverage the knowledge they have already gained on the levels 1 - 4. Users can create applications using predefined customizable components (level 1). Components and pages have WYSIWYG-like representations (level 2), which reveal the contents. In XIDE visual editor (level 3), presented in Figure 2, users visually add components on the page and connect them. XIDE supports direct manipulation for components,

[7]XIDE, code.google.com/p/xformsdb-ide

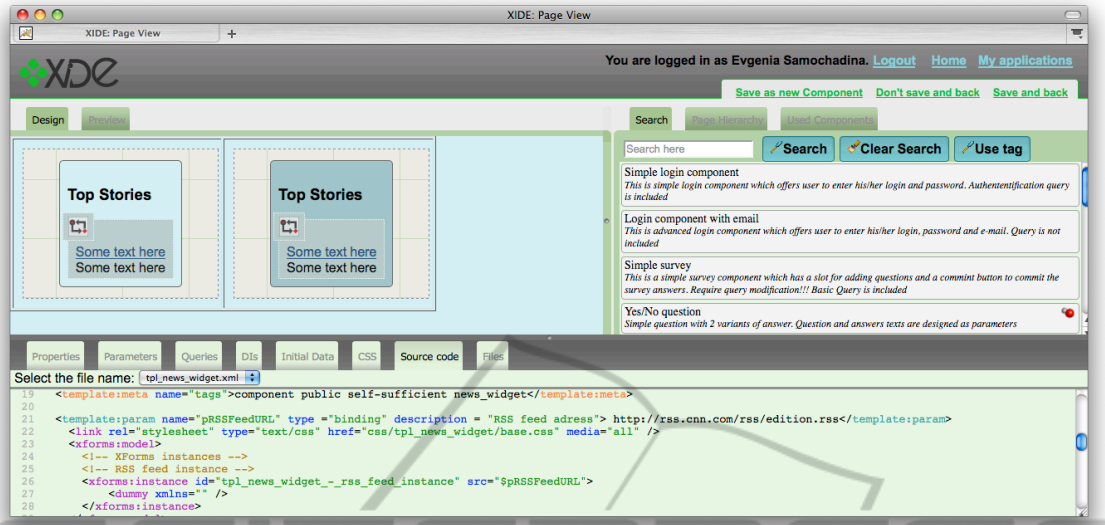

Figure 2: Editing a page in XIDE.

i.e., drag-n-drop for adding, managing, and deleting components in order to lower getting started barriers. Users can edit the source code of each component, written in a markup based language (level 4).

Second, XIDE provides an advanced source code editing functionality. The source code is written in the markup based framework, so editing activity belongs to the level 4. The framework is unified, so it allows to build interactive Web applications, including server-side and database functionality. Hence, users can achieve the tasks from the level 5. Moreover, users can incrementally learn the technology and become capable of editing more advanced code (level 6). XIDE provides wizards and an intelligent automatic generation of source code for configuring technical issues that can be complicated for end users. The built-in text editor supports advanced code highlighting and error checking. Syntax highlighting aims to focus attention of users to the most important parts of the source code, e.g., parameters.

Finally, XIDE reveals the connection between visual and source code editing activities, which is a challenging issue for end users (Ko et al., 2004). XIDE uses multiple coordinative windows to show one concept from different perspectives or different degrees of abstraction. That allows end users to discover the link between, e.g., component source code and component visual output. XIDE also supports *design at runtime*: changes made in the source code of the page or component instantly appear in design or preview representation.

XIDE has four views for different development activities: getting started, list of applications, application management, and page editing. Each view contains information and functionality related to the activity at hand.

Extending the approach of (MacLean et al., 1990), XIDE provides several levels of modification, such as component customization, visual manipulation, page source code modification, component source code modification, etc.

4.2 Background Technology

In XIDE, XFormsDB[8] (Laine et al., 2012) is used as background technology for both components and applications. XFormsDB is a comprehensive Web application development and hosting platform that allows users to implement interactive, database-driven Web applications using only markup languages. It is based on the XForms[9] markup language and its extensions that specify common server-side and database-related functionalities.

The presentation-centric approach of XFormsDB offers an attractive solution for users with little technical knowledge (Laine et al., 2011). For users with the skill level 4, this selection is even more promising, because they are already familiar with HTML and the concept of markup languages. Thus, XIDE meets the requirements R1 and R2.

In order to fulfill requirement R3, we designed an intermediate markup language, called Template Language (TL). TL is used to support the component-based architecture, i.e., define components and build them into Web pages. TL was designed as a XML-

[8]XFormsDB, code.google.com/p/xformsdb

[9]XForms standard, http://www.w3.org/TR/xforms11

based language in order to natively build TL into XFormsDB and utilize the same concepts.

Except the TL, there is no internal API or library that a developer is forced to use while editing existing or creating new components. That makes the modification and contribution of new components a straightforward task, because it requires minimum additional skills and knowledge of technology (Lieberman et al., 2006). Thus, XIDE meets requirement R3.

5 EVALUATION

We conducted a preliminary user study to observe how the level 4 users perform editing tasks related to level 5 and 6 in XIDE.

In total, there were 9 participants: 5 male and 4 female. Participants were recruited from Economics and Computer Science students. We limited the background of participants to the level 4. They were active Internet users, had tried to create a simple Web page using a visual editor or tried to use some XML-based markup language. None of the participants studied Web development as a major. None of them had ever used XIDE, XForms or XFormsDB.

For the study, we used usability evaluation method, combined with thinking aloud and semi-structured interviews. Each evaluation took approximately 1.5 – 2 hours.

Each participant was asked to perform predefined tasks, which were designed to investigate the transition from visual editing to source code editing in XIDE. Briefly, the participants were asked to: modify the application using a visual editor and manually in the source code; modify the page source code; do minor and major modifications to TL and component logic; and contribute new components. For example, participants were asked to introduce a second parameter to a RSS Reader component and use the modified component in the code.

5.1 Results and Discussion

Our hypothesis was that users often need to edit the source code when they are using a visual tool. Moreover, they could start editing the source code if they had some minor skills in the background technology and if they were facilitated to do so. The results of the study showed that participants were able to complete all the tasks related to code editing. Participants were able to understand the concepts of the TL and modify it. In general, participants said they found the approach implemented in XIDE promising and would use such kind of a tool to solve the problems occurred at work. They found it useful and flexible, because it supported different levels of modification.

During the interviews, participants reported they have previously faced problems due to narrow scope of existing visual tools. Those problems forced them to switch from visual modification to direct code editing. Their experience showed that no visual approach can cover all possible needs, and thus direct editing is required anyway. Some participants complained about difficulties they experienced previously while editing auto-generated source code produced by WYSIWYG tools for HTML editing.

While performing tasks in XIDE, participants widely employed the copy-paste approach. XIDE does not provide any vocabulary of XFormsDB or HTML tags that could be used or any code completion. Thus, the only way how a novice user could contribute a piece of new code was to find the existing code that did relatively similar thing, copy it, and modify it according to the task. Two main observations of this study were that users natively intended to use the copy-paste approach and were able to use it successfully with XFormsDB.

During those copy-paste activities, an immediate response from the system was highly appreciated. It allowed experimental programming, in which users tried different options to see which option lead to the desired result.

XIDE advanced source code highlighting is a powerful tool. Besides highlighting markup language tags and syntax, it emphasized TL structure and user interface elements. We had a hypothesis that novice users most likely would edit the code related to component management and representation of the information. Therefore, we used grey color to dim less important areas, such as a data model and a header. Participants focused on code that most likely contained the needed information. Yet, all code was visible, so participants could check it if needed. All users named this feature among the most usable features in XIDE.

Users generally were able to *read* the code, because they knew the concept of tag and guess the functionality of unknown tags based their names. However, some of the tag names were not self-descriptive enough. This issue should be taken into account during future redesign of languages, e.g., the next versions of XFormsDB and HTML. *Id*s, which developers widely use in the source code to address XML structures, should be more descriptive and close to the natural language.

Another issue users complained about was the lack of debug information and error descriptions. Although users were satisfied with the on-time preview feature and error code highlighting, they asked for

more descriptive feedback from the system when an error occurred.

The highlight approach used in XIDE can be expanded to have different code highlighting schemas for different types of tasks users perform. For example, managing parameters or changing user interface.

Generally, the component-based approach implemented in XIDE, together with features that facilitate source code editing can be used for education purposes. A person who has a superficial knowledge of some technology can use a tool for a period of time for self-education (Zang et al., 2008). While a person uses a tool, he or she gains practical knowledge and can increase skills relatively gently, without having big educational barriers.

6 CONCLUSIONS

In this paper, we addressed the problem of expanding end-user Web development possibilities. We classified end-user Web development activities into levels, and addressed the problem of smoothing the transition from the level where end users use visual tools to the next one, where they can edit the source code. Our main contribution is a combination of a component-based visual tool with a unified, markup-based background technology. Our approach allows end users to leverage the knowledge they already have to create more advanced Web applications with server-side functionality.

In our proof-of-concept implementation, we designed and implemented a visual component-based tool called XIDE. XIDE is designed to support visual modification and especially direct code editing with different features, such as code highlighting and immediate previewing. We used XFormsDB as a background technology. It is a recently developed tier-expanding framework based on markup languages. End users, who have a basic knowledge of HTML, can intuitively guess the meaning of the source code written in XFormsDB. The XIDE prototype was preliminary evaluated by means of a user study, in which users were able to complete all tasks successfully. Thus, the proposed approach enables users to make changes in the source code and expands users possibilities in Web development.

REFERENCES

Cappiello, C., Matera, M., Picozzi, M., Sprega, G., Barbagallo, D., and Francalanci, C. (2011). DashMash: a mashup environment for end user development. *Web Engineering*, 6757:152–166.

Cardone, R., Soroker, D., and Tiwari, A. (2005). Using XForms to simplify Web programming. In *Proc. of the 14th international conference on World Wide Web*, WWW '05, pages 215–224. ACM.

Costabile, M. F., Mussio, P., Parasiliti Provenza, L., and Piccinno, A. (2008). End users as unwitting software developers. In *Proc. of the 4th International Workshop on End-User Software Engineering*, pages 6–10. ACM Press.

Cypher, A., Lau, T., Nichols, J., and Dontcheva, M. (2009). Workshop on end user programming for the web. In *Proc. of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems*, pages 4779–4782. ACM Press.

Grammel, L. and Storey, M.-A. (2008). An end user perspective on mashup makers. In *University of Victoria, Tech. Rep. DCS-324-IR*.

Ko, A., Myers, B., and Aung, H. (2004). Six learning barriers in end-user programming systems. In *Proc. of the 2004 IEEE Symposium on Visual Languages-Human Centric Computing*, pages 199–206. IEEE.

Ko, A. J., Myers, B., and Rosson, M. B. (2011). The state of the art in end-user software engineering. *ACM Computing Surveys*, 43(3):21.

Laine, M., Shestakov, D., and Vuorimaa, P. (2012). Extending XForms with Server-Side Functionality. In *Proc. of the 27th ACM Symposium on Applied Computing (to appear)*.

Laine, M. P., Shestakov, D., Litvinova, E., and Vuorimaa, P. (2011). Toward Unified Web Application Development. *IT Professional*, 13(5):30–36.

Lieberman, H., Paterno, F., Klann, M., and Wulf, V. (2006). End-user development: An emerging paradigm. *End User Development*, pages 1–8.

MacLean, A., Carter, K., Lövstrand, L., and Moran, T. (1990). User-tailorable systems: pressing the issues with buttons. In *Proc. of the SIGCHI conference on Human factors in computing systems: Empowering people*, pages 175–182. ACM.

Mosemann, R. and Wiedenbeck, S. (2001). Navigation and comprehension of programs by novice programmers. In *Proc. of the 9th International Workshop on Program Comprehension*, pages 79–88. IEEE.

Park, T. H. and Wiedenbeck, S. (2010). First Steps in Coding by Informal Web Developers. In *Proc. of the 2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 79—-82. IEEE.

Repenning, A. and Ioannidou, A. (2006). What makes end-user development tick? 13 design guidelines. *End User Development*, 9:51–85.

Rode, J., Bhardwaj, Y., Pérez-Quiñones, M., Rosson, M., and Howarth, J. (2005). As easy as Click: End-user web engineering. *Web Engineering*, 3579:478–488.

Rosson, M. B., Ballin, J. F., Rode, J., and Toward, B. (2005). Designing for the Web Revisited : A Survey of Informal and Experienced Web Developers. *Web Engineering*, pages 605–620.

Won, M., Stiemerling, O., and Wulf, V. (2006). Component-based approaches to tailorable systems. *End User Development*, pages 115–141.

Zang, N., Rosson, M. B., and Nasser, V. (2008). Mashups: who? what? why? In *Proc. of the CHI'08 Extended Abstracts on Human Factors in Computing Systems*, pages 3171–3176. ACM.