University of Eastern Finland

Faculty of Science and Forestry, School of Computing

Department of Computer Science and Statistics

International Master's Degree Programme in Information Technology

Evgenia Litvinova

XIDE - a Visual Tool for End User Development of Web Applications

Master’s Thesis

Joensuu, Finland, 2010

Supervisors:	Professor Petri Vuorimaa
	Professor Jussi Parkkinen
Instructor:	Mikko Pohja

Abstract

Because of increasing popularity of Internet and Web 2.0, computer users become more familiar with the concepts and possibilities of the Web, such as markup languages and mashups. They become more and more involved in creating of Web application for solving their personal tasks. Modern end users often have wider development skills and experience then pure end users in traditional understanding; general level of users knowledge has increased and they perform some development activities. For the development, end users can choose either a tool created for professionals or special end user Web development tool, which allow to create simple applications using visual interface, but force users to use complex technologies if they want to achieve more complex task.

The aim of the Thesis is to create a tool that has a gentle slope of complexity. Using such a tool, each end user can choose the level of complexity that suits desired task and skills of concrete person. It also facilitates leaning, since the effort user needs to spend is in proportion to the complexity of task that needs to be solved.

The idea described in this Thesis is to combine component-based approach with XFormsDB Web application development framework. XFormsDB framework allows to create an interactive Web application using fully declarative approach and thus reduces the difficulties related to learning complicated technologies and concepts and combining them together.

The requirements for the tool functionality are constructed based on analysis of end users challenges, tasks and experience related to Web development. A review of related scientific research and existing commercial tools are made to analyze common approaches used in end user Web development tools. During tool design process, successful approaches and user interface decisions are combined in order to employ gentle level of complexity and facilitate end users in Web development. The working tool prototype called XIDE is implemented and evaluated by means of usability testing, creating sample applications and expert evaluation.

Keywords: end user, end user development, end user programming, Web application, end user Web development, gentle slope of complexity, reuse, component-based approach, XFormsDB

Acknowledgments

I would like to thank the following people:

- Professor Jussi Parkkinen for giving me a chance to participate in IMPIT program and do my studies in Finland, for being my supervisor and thus making this research possible
- Roman Bednarik for being the coordinator of IMPIT program and for opening my eyes to the world of usability
- Professor Petri Vuorimaa for offering me the opportunity to do this Thesis work in Aalto University, and for all his guidance, ideas and comments he gave me while being my supervisor
- Members of Web Services research group, and especially Pia Ojanen, for their feedback about the text and the whole theme of facilitating writing. Without them this Thesis would have never reached the level it has now
- Mikko Pohja for his comments and suggestions
- Markku Laine for all his advises and practical help and for being an example for me
- My family and friends for their background support and continues questions regarding the date when I finish the Thesis
- My husband Andrey Litvinov for his support and all-round help. I would never finish this Thesis without you!

This research was done in Aalto University as a part of TIVIT's Flexible Services program and its Ecosystem Design and Evolution (EDEN) project, funded by National Technology Agency of Finland (Tekes) and Nokia Research Center (NRC).

Espoo, December 2010

Evgenia Litvinova

samochadina@gmail.com

Table of Contents

Abbreviations and terms

AJAX	Asynchronous JavaScript and XML
CSS	Cascading Style Sheets
EU	End User
EUD	End User Development
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IT	Information Technology
Java	An object-oriented programming language
JavaScript	A scripting language
MVC	Model-View-Controller architecture
TL	Template Language
POST	An HTTP request method, used by the client to send data to the server
RIA	Rich Internet Application

RPC	Remote Procedure Call
UI	User Interface
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
Web	World Wide Web
WWW	World Wide Web
WYSIWYG	What You See Is What You Get user interface paradigm
XML	Extensible Markup Language
XForms	An XML application representing the next generation of forms for the Web
XFormsDB	An XForms-based framework for simplifying Web application development
XIDE	XFormsDB IDE
XPath	XML Path Language
XQuery	XML Query

List of figures

List of tables

1. Introduction

1.1. Introduction to a problem

Nowadays, computers and Internet increasingly spread into lives of ordinary people. Specialists from different areas of research and industry, students, office staff and other non-professionals use computers to solve their daily problems and tasks. In addition to simply using a ready-made software and browsing the Internet, people have started to participate in development activities even without knowing that they are actually doing the development, e.g., writing macros, contributing to wikis using markup languages for text formatting, writing rich posts with media elements in their personal blogs. (Costabile, Mussio, Parasiliti Provenza, & Piccinno, 2008) General computer knowledge and some development skills are required at many working positions that are not linked to professional programming. (Ko, Myers, & Aung, 2004) According to predictions described in (Scaffidi, Shaw, & Myers, 2005), in 2012 in US there will be only 3 million professional programmers and more than 55 million people who do some development activities, including using spreadsheets and databases.

The majority of non-professional development activities are related to Internet and Web technologies. For instance, people create personal pages in WYSIWYG (What You See Is What You Get) editors, write search expressions, mash up services and data, create Web surveys in special tools, etc. Web development became so popular because Web provides a good ground for communication, collecting the information and sharing it with others. (Rosson, Ballin, & Rode, 2005) This motivates people to learn how to contribute to the Web by means of some software development technologies. Moreover, the general level of computer knowledge has increased, so people consider feasible some tasks and usage of technologies and tools they were previously confused about. (Cypher, Lau, Nichols, & Dontcheva, 2009)

The tendencies described above motivated non-professional computer users to shift from *pure end users*, people who only use the ready-made software, to *end users* or *end user developers*, people who do some programming activities to achieve their own goals. End user developers may occasionally have high degree of knowledge in some related field, but they are apparently interested not in programming itself, but in creating software artifacts that help them in their daily life activities. (Costabile, Mussio, Parasiliti Provenza, & Piccinno, 2008) Even though end users cannot be considered as completely inexperienced users anymore, tools and technologies created for professional developers are often not suitable for them. Powerful professional tools are aimed at producing complex results with versatile functionality and thus have high threshold of necessary knowledge to start the development. (Myers, Hudson, & Pausch, 2000) (Costabile, Mussio, Parasiliti Provenza, & Piccinno, 2008) Often end user developers have relatively small tasks to solve and thus learning how to use and using a complex professional tool or technology requires too much effort.

Also, there is a huge gap between using markup language for inserting picture in a blog post and manually coding a full Web page using HTML. In last fifteen years, Web applications have evolved from static text to dynamic and highly interactive applications. (Jazayeri, 2007) However, several years ago end users mostly were able to create static pages only. (Rode, Rosson, & Quinones, 2006) Development of interactive applications is more complicated, because it involves using of several technologies and combining them together. As the result, today complex interactive Web applications are still mostly created by professional developers.

End user Web development research area, which is increasingly active in recent years (Cypher, Lau, Nichols, & Dontcheva, 2009), aims to allow end users to create interactive Web applications themselves. Two general approaches are used to overcome difficulties of Web programming: by making programming languages more transparent and understandable for end users or by providing tools that allow users to achieve the result without actual programming. (Kelleher & Pausch, 2005), (Cypher, Lau, Nichols, & Dontcheva, 2009)

One group of tools has a narrow focus on a special type of resulting applications, e.g., online survey, blog or mashup. The tool can provide workflow that guides end user through wizards, visual modifications, and keeps user on high abstraction level in order to eliminate programming details. (Grammel & Storey, 2008) Generally, the tool allows user to create working software artifact without any coding. However, if the demanded application functionality is beyond the predefined scope of the tool, it becomes hardly possible for the end user to complete the application development. The tool can support manual modification of the underlying source code, but that immediately requires from the end user advanced knowledge of specific concepts and complex technologies. (Zang, Rosson, & Nasser, 2008)

The problem of those tools is that they force user to overcome high barrier of complexity when direct edition of the application is required. Common trend in the end user development is providing tools with gentle slope of complexity (Lieberman, Paterno, Klann, & Wulf, 2006), so that the effort user needs to spend is in proportion to the complexity of task that needs to be solved. When using such tool, small increase in task complexity results in small effort to learn new functionality. This approach supports a range of tasks and skills of the user (MacLean, Carter, Lovstrand, & Moran, 1990) and assists in growing of their knowledge and needs. (Rode, Rosson, & Quinones, 2006)

The tools with wider scope often employ component-based approach in order to provide gentler slope of complexity as well as employ reuse strategy. (Rode, Bhardwaj, Perez-Quinones, Rosson, & Howarth, 2005) End users can add, customize and combine predefined components. Thus, they can create an application with higher functionality that they can attain by doing development from scratch. (Hartmann, Doorley, & Klemmer, 2008) However, if there is no appropriate component, an existing one needs to be modified or adjusted to the user's task. Users have to lower the abstraction level they use and face the source code of the component, which requires from them knowledge of system architecture and the background technology of the component.

Besides visual tools that eliminate programming difficulties, another option can be engaging users in actual programming by lowering programming barriers and providing more appropriate technologies. (Kelleher & Pausch, 2005) Two most important difficulties of programming for Web for end users are unknown concepts and different technologies that have to be combined together. There are two directions in simplifying of programming: simplification of the language itself or reduce the amount of languages to integrate and the effort on combining them.

Since end users are becoming familiar with markup languages by contributing to wikis and personal blogs (Myers & Ko, 2009), HTML can be considered a good option for basic technology for the end user Web development. HTML belongs to the family of declarative languages that have higher level of abstraction and thus are claimed to be more convenient for non-professional users. (Honkala, 2006) However, single HTML only supports creation of static pages. In order to achieve interactive Web application, it should be combined with other scripting and server-side technologies. (Ramirez & Wroblewski, 2005) (Jazayeri, 2007)

In this Thesis, it is proposed to use XFormsDB (Laine, 2010) declarative framework developed at Aalto University. It is an extension of XForms, W3C standardized technology for creating interactive Web forms. With XFormsDB it is possible to create a Web application by writing only client-side code using declarative languages, which reduces the number and complexity of technologies to learn and combine. XFormsDB is claimed to be convenient for authoring highly interactive Web applications by non-professionals.

This Thesis addresses the problem of creating visual tool called XIDE for developing interactive Web applications by the end users. The tool aims to achieve a gentle level of complexity by extending visual component-based approach, described in (Ko, et al., 2009) (Won, Stiemerling, & Wulf, 2006), (Rode, Bhardwaj, Perez-Quinones, Rosson, & Howarth, 2005) and combining it with XFormsDB framework (Laine, 2010) as a background technology for components. The tool provides a wide range of levels of modification complexity in order to allow users to select the level that is appropriate to their tasks and level of knowledge.

1.2. Research Objectives and Scope

The first objective of this Thesis is to design a way how to combine component-based approach and XFormsDB framework in order to create a visual tool that supports gentle level of complexity and facilitates end users in creating interactive Web applications. The tool should combine existing solutions from end user research and commercial sector in order to support end user developers with wide range of skills in creation of Web applications with different level of complexity. The second objective is to implement a working Web-based prototype tool called XIDE that employs the design and examine the tool by means of usability studies, expert evaluation and implementing test applications.

Because of limited resources of the project, some restrictions are imposed on the scope of this Thesis. No empirical study or survey is conducted to gain information about problems, barriers and expectations of end users, however this information is received from previous scientific publications and analyzed in the background part. Analysis of tasks of end users and designing a sufficient set of relevant components are also out of scope of this Thesis and considered as future work. Finally, evaluation of the developed prototype tool is limited to check the feasibility of approach and make a decision about continuation of the work.

1.3. Research questions

Since end user development is an active research and practical area, there are many tools and solutions that aim to help end users in development of Web applications. Visual component-based approach, which is actively used, supports manipulations on high abstraction level and allows users to create an application without actual coding. However, the user is required to overcome a difficulty of learning complex technology and the whole architecture of the system in order to be able to change a component or contribute a new one.

XFormsDB technology is promising in a context of simplifying Web application development; it was designed considering needs of non-programmer users. (Laine, 2010) It supports interactive Web application creating using only declarative languages and thus lowers the starting threshold and smoothes the learning curve.

The initial idea of this Thesis is to combine those two approaches to create a tool that facilitates end user development of Web applications and supports gentle slope of complexity.

Question 1: How to combine component-based approach with XFormsDB framework in order to design a tool that supports gentle slope of complexity?

Even though general approach is already defined in the scope of this Thesis, there are still many details to investigate. In (Won, Stiemerling, & Wulf, 2006) authors raise a question whether component-based approach should be combined with other interaction techniques in order to achieve gentle slope of complexity and how to design such a tool. The following questions form to the main research question:

What user interface approaches should be utilized to facilitate the development of Web applications? How to design component model? How to design the architecture of component reuse?

In order to answer to the main research question, the prototype tool called XIDE is designed, implemented and evaluated. Two additional research questions are investigated to support the main research question.

Question 2: What features should the tool provide to facilitate end users development?

This question contains the following questions:

What tasks and level of skills do end users have? What are technical problems end users faces during Web development? What are the expectations of end users about the tool that they use for development of Web applications? What functionality should it provide? How do end users evaluate existing tools? What design solutions do they find useful?

Answers to those questions are gained from review of publications, which report results of user surveys and user experience in using of existing tools. The requirements for the tool are based on the answers to this research question.

Question 3: What are common user interface approaches used in end user development tools?

Review of existing approaches and solutions provides a background understanding of successful and failed approaches and common problems of those solutions, which is essential for creating a new solution. (Myers, Hudson, & Pausch, 2000) The following questions are raised in scope of this question:

What are the existing solutions for the problem of end user development for Web in research and commercial sector? What are advantages and drawbacks of those solutions? Do those solutions support gentle slope of complexity? What features from those solutions should be utilized in XIDE?

The answers to those questions are based on review of scientific publications about state of the art in end user development research area and review of existing commercial tools for end user Web development. Conceptual and user interface design of the tool are based on the answers to this research question.

1.4. Research methods

The main research method used in this Thesis is constructive research, also known as design science. (Järvinen, 2004) The constructive research method is natively used in the computer science domain to produce practical solution for relevant problems and theoretical contribution to the research domain, which owns the problem. According to Järvinen, this research method is used when a new innovative solution needs to be invented or built.

This research method is used to achieve the answer to the main research question, described in Section 1.3. The practical solution is the prototype tool for solving the problem of end user development of Web applications and support of gentle slope of complexity. As it is defined by constructive research, the work begins with deep study of the problem and the research domain. (Crnkovic, 2010) This research is described in Chapters 2 - 4. After that, the solution is invented, designed and implemented (Chapters 5 and 6). Implementation of the solution is the practical contribution of this research. In Chapter 7, the developed prototype tool is evaluated and validated by means of *experience*, *evaluation* and *example* (Shaw, 2003): observing user experience using usability tests, expert evaluation and example use case. Theoretical contributions are analyzed in conclusions in Chapter 8.

1.5. Organization of the Thesis

This Thesis is organized as follows. First part of the work consists of literature review. The next Chapter introduces the concepts of end user and end user development and discusses what are the skills, needs and problems of modern end users. Chapter 3 presents review of state of the art in end user development of Web applications research area and explores related solutions. The second part of Chapter 3 presents an overview of existing commercial tools for end user development and discusses end user experience of those tools. Finally, in the conclusion of the Chapter 3 there is a summary of the analysis of different commercial tools. Chapter 4 briefly describes the XFormsDB technology and explains why it is appropriate selection for the work described in this Thesis. It also briefly discusses what challenges user face during development of XFormsDB-based applications from scratch. Together Chapters 2 - 4 form the background for the further work. Chapter 5 summarize the analysis of end user problems and presents the both functional and technical requirements for XIDE based on those problems.

Second part of the work describes design, implementation and evaluation of the tool. Design of component-based architecture and user interface is described in Chapter 6. It also presents implementation details. Chapter 7 starts with evaluation of XIDE using usability testing with real users that was performed to examine the approach in general and improve the design. It also discusses the evaluation of the XIDE tool by a mean of expert evaluation and by presenting the process of creating sample Web application. Finally, Chapter 8 presents conclusions, contributions of this Thesis and future work.

2. End Users

Background of the work described in this Thesis is related to several complementary domains of research and practice: Web engineering, end-user development, mashups, component-based development and XFormsDB framework. This Chapter introduces theoretical background about development of Web applications by end users. It gives definition of end users, discusses their tasks and level of knowledge, analyses problems and challenges they face during Web development and their expectations of using a Web development tool. Finally, this Chapter introduces the concept of gentle slope of complexity and explains why it is a promising approach to be used in end user development.

2.1. End user developers

According to (Myers & Ko, 2009), end user development is called a promising area of research and practice. Concepts of end users, end user programming and end user Web engineering refined in (Ko, et al., 2009). According to that, *end user* is a simple computer user, who initially was supposed only to operate with software artifacts, not to contribute new functionality. However, nowadays popularity of programming increased and is almost the same as of simple usage of a computer. Most of users perform some customization or programming, however some of them are not aware of the fact that they are actually programming. (Costabile, Mussio, Parasiliti Provenza, & Piccinno, 2008) According to (Ko, et al., 2009), *programming* is a process of writing a set of specifications that can be executed or interpreted by computer. *End user programming* is a process of programming to achieve the result, which will be used mostly personally. Professional programming, in contrast, aims to develop artifacts to be used by other people. End users, same as professionals, meet the quality issues during development of software artifacts.

There is another modification of end user definition that can be found in papers, e.g. (Costabile, Mussio, Parasiliti Provenza, & Piccinno, 2008), (Myers, Ko, & Burnett, 2006), and that is going to be used in this Thesis. *End user* is a person who does programming not for work purpose, but to accomplish some personal goal. Because end users are not professionals, they are not interested in the technology and programming process as such. They are focused on achievement of concrete tasks, which appeared from their personal life, and do programming only if they believe that can help them in their everyday activities. (Floyd, Jones, Rathi, & Twidale, 2007)

According to (Ko, et al., 2009), it is important to not confuse end user with *inexperienced users*, who does not have any knowledge of certain field or technology, or *novice users,* who have just started to learn how to use new technology or tool. However, novice or learning users are respectively close to the end users, because of self-learning nature of end users. (Myers, Ko, & Burnett, 2006)

End users cannot be considered as a uniform group of people; they have different goals, tasks and activities. (Costabile, Fogli, Fresta, Mussio, & Piccinno, 2004) People have different background and have different interaction habits of using a computer. As stated in (Ye & Fischer, 2007), traditionally end users and developers are considered as two mutually exclusive groups: user has a problem to solve, developer constructs software artifact to solve it. However, nowadays users are more and more involved into actual developing in order to solve their problem by themselves. In (Costabile, Mussio, Parasiliti Provenza, & Piccinno, 2008) this idea was refined in details (Figure 1). They classify users according to variety of programming activities they perform: *pure end users*, who do not do any programming; *end users who perform customization activities* to adopt the existing object to their preferences, such as changing colors and other parameters or managing toolbars; *end users who write macros* in order to automate some operation; *end users who develop Web applications*, who might have narrow skills of certain technology or in usage of certain tool, but have poor knowledge in computer science; *developers who use domain-specific languages* are professionals in some domain and are able to develop software artifacts to solve the problem of the domain; *data-intensive researchers* can be considered as almost professional programmers, however they do not have any software engineering background; finally, *professional programmers* close the classification. The category to which a user is assigned at the moment depends on the task and user's skills. For example, a user can choose lower category if it is enough to complete the task or upper category if user acquired new skills and can now accomplish more advanced tasks.

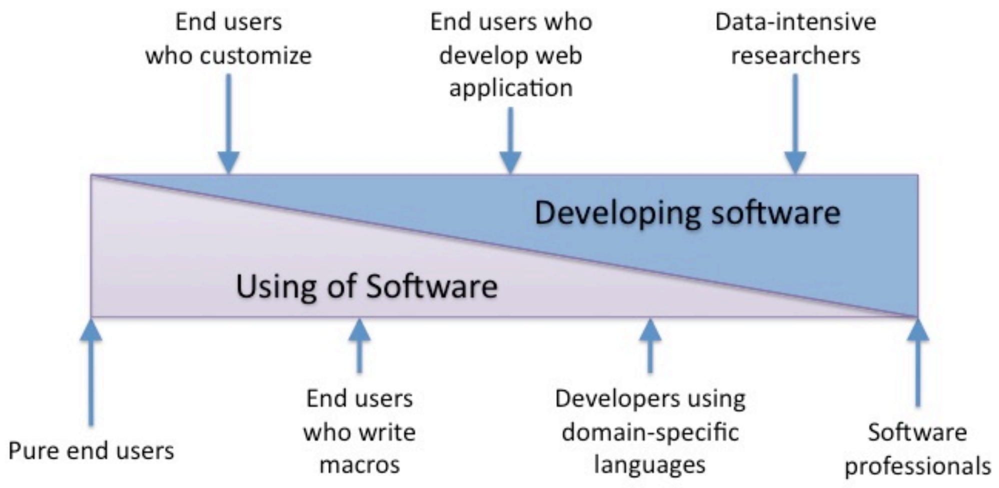

End users who customize
End users who develop web application
Data-intensive researchers
Developing software
Using of Software
Pure end users
End users who write macros
Developers using domain-specific languages
Software professionals

Figure 1 The specter of software-related activities, adapted from (Costabile, Mussio, Parasiliti Provenza, & Piccinno, 2008)

End users who are close to pure end users category often are not aware that they are really do some programming, however they do have certain skills and experience and thus are *unwitting developers*. Going from pure end users to professional programmers, end users become more and more skilled and knowing what they are doing. (Costabile, Mussio, Parasiliti Provenza, & Piccinno, 2008)

2.2. End user Web developers

Summarizing publications about variety of end user Web development activities and tasks (Rosson, Ballin, & Rode, 2005) (Zang, Rosson, & Nasser, 2008) (Rosson, Ballin, & Nash, 2004), the idea of user classification can be employed to Web development domain individually (Figure 2). The following classes of end users can be identified: *pure end users*, who do only Web browsing; *end users who customize* their personal homepage or blog by setting parameter values; *end users who use visual editors* to create custom information gathering application, such as survey (Rode & Rosson, 2003), static HTML page or a mashup without any actual programming (Zang, Rosson, & Nasser, 2008); *end users who use markup languages*, e.g., to contribute to wiki or write advanced post to their blog (Myers & Ko, 2009); *end users who are familiar with some programming language* and use it to achieve advanced task or modify existing functionality in the tool that provides visual environment and enables direct modification of the source code; *professional end users* or *informal Web developers* who are not professional programmers (Rode, Rosson, & Quinones, 2006), but have knowledge of several complementary technologies and are able to construct interactive client-server Web application for personal usage; *professionals* who does Web development for work. The complexity of activities, which are performed by each group of users, increases from left to right.

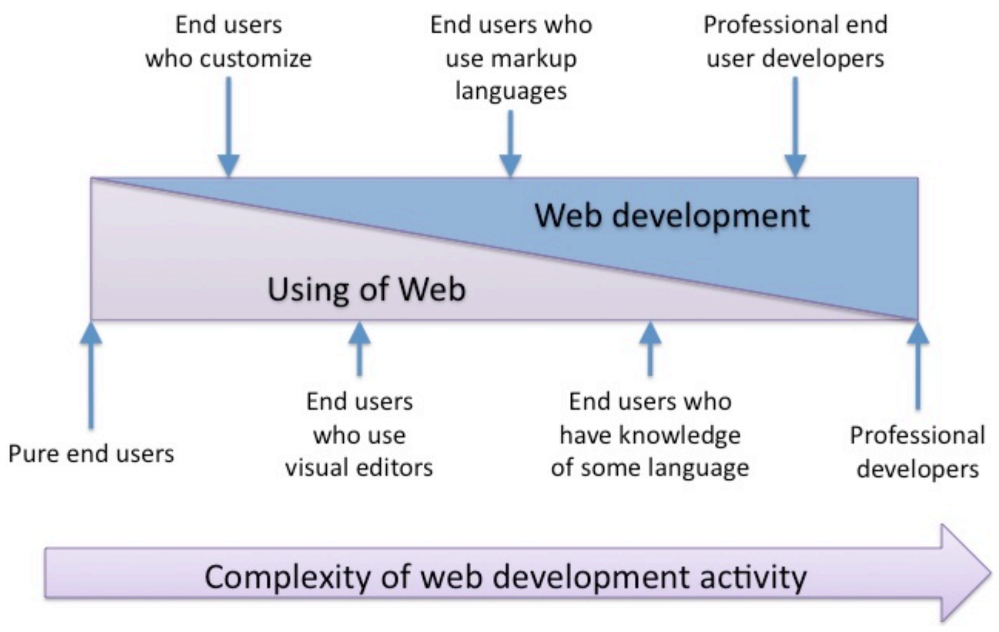

End users who customize
End users who use markup languages
Professional end user developers
Web development
Using of Web
Pure end users
End users who use visual editors
End users who have knowledge of some language
Professional developers
Complexity of web development activity

Figure 2 The spectrum of activities related to Web development

Going over from left to right, user groups can be considered as evolution steps for the user who passes from pure end user to professional. In case if one technology is used through the whole evolution process, the upgrade on the next level is easier for the user because the knowledge gained on the previous one. Since many users can be referred as users who use markup languages, declarative languages, such as HTML and XML, are good option to consider as basic technology.

In end user development for the Web, the gap between professional end users and previous group is really difficult to overcome because of the nature of Web. Interactive Web applications usually consist of client and server parts, which require knowledge of different languages and paradigms and combining them together.

2.3. Development of interactive Web application

(Myers, Hudson, & Pausch, 2000) claim that Web has become so popular because everybody can contribute own applications to it. In scope of this Thesis, Web application term means an application, which is accessed using Web browser program through the Internet. Web application utilizes client-server paradigm. Web browser acts as a client, which connects to remote computer, where Web application is deployed (a server). Initially Web application term meant dynamic Web site, in contrast to static Web site. However, nowadays Web site and Web application terms are often used as synonyms.

In contrast to desktop applications, there are two main advantages of Web applications. First is the fact that a Web application can be accessed from any computer, which has a browser installed and working Internet connection. Generally, Web applications are cross platform, what means that one Web application can be accessed from different operating systems. Second advantage is that Web application can be updated only on the server, so all clients can access new version of the application or see new information instantly. Because of these advantages, Web applications are widely used for information sharing by everybody, including non-professional users.

Initially Web applications were static text pages for scientific purposes, written in HTML language. HTML is a markup language, which provides means for creating static structured Web page with paragraphs, headers and links. (HTML 4.01 Specification, 1999) It is also possible to include user interface controls, such as input fields and buttons to develop a Web form. Client-side interactivity can be achieved by having several pages showed to user in a sequence.

Most of current Web applications are highly interactive and provide functionality comparable to a desktop application. (Jazayeri, 2007) Notion of interactive Web application means the Web application, that does not consist only of static pages, but contains forms, animations, provides immediate feedback, interacts with user without reloading the Web page. Interactive applications are mostly created by professionals using professional languages, because of the complexity of Web development described below.

There are many different technologies appeared to support creation of interactive Web applications. (Fraternali, 1999) (Ramirez & Wroblewski, 2005)(Jazayeri, 2007) These technologies can be classified into client and server technologies. Client technology (e.g., CSS, JavaScript, XForms, Flash, Java Applets) assumes that the browser processes the application interactivity. Using server technology (e.g., PHP, ASP, JSP, etc.) the application is executed on the server and provides results to the client browser. On practice, client and server technologies are usually combined in order to create interactive user interface and implement application logic on a server. One traditional approach is to have HTML and JavaScript on client-side and Java on the server.

Thus, in order to create interactive Web application developer has to combine several client and server technologies. This makes development process complicated and requires more knowledge from the developer. However, using fewer technologies or even one programming language could ease the development and client-server communication. (Kuuskeri & Mikkonen, 2009)

Another approach to reduce complexity of Web development is simplifying the technologies or languages being used. Generally, languages can be separated into imperative, declarative and mixed. Imperative languages describe *how* the program should perform the task, while declarative languages specify *what* task the program should perform.

Although imperative languages are more powerful, declarative languages traditionally have higher level of abstraction and thus are claimed to be more useful in case of non-professional users. (Honkala, 2006) Available server technologies are mainly imperative, however there are several declarative client technologies that support development of highly interactive applications. (Laine, 2010) As it was discussed in previous Chapter, this Thesis proposes to use XFormsDB framework, which allows to create a Web application by writing only client-side code using declarative languages. XFormsDB is claimed to be useful for authoring highly interactive multi-user Web applications by non-professionals. More information about the framework can be found in Chapter 4.

2.4. End user development and end user software engineering

Two research domains were established to satisfy two initial polar groups, pure end users and professional developers. *Software engineering* research focuses on professional developers and covers process of development of software artifacts that will be later used by end users as a solution for their problem. *End user development* research focuses on techniques and tools that can help end users to solve their problem by themselves. (Ye & Fischer, 2007) End user programming domain focuses on needs and problems of the users who do actual programming, while end user development field is more wide and covers all activities that end user perform to solve their problem using a computer. (Repenning & Ioannidou, 2006)

Traditionally, solutions developed by those research domains are either aimed at the pure end users or professional developers. For example, analysis of past user interface tools showed the gap between easy to learn tools and tools with wide set of possible tasks to achieve. (Myers, Hudson, & Pausch, 2000), According to the analysis, the threshold of the system is how difficult is to learn how to use it and the ceiling is how much you can do with the system. Most of successful systems either have low threshold and low ceiling or high threshold and high ceiling. Problem to create the system with low threshold and high ceiling still remains unresolved.

End user software engineering is defined as systematic and disciplined activities that are aimed to construct a high quality software artifact. However, because the main focus of the end user is to achieve the goal, activities related to design, debugging, testing, verification and maintenance are secondary to the main goal of development the working program. (Ko, et al., 2009) Moreover, end users often do not have strong background in computer science. As a result, end user software engineering has opportunistic, ad hoc nature. (Hartmann, Doorley, & Klemmer, 2008)(Cao, Riche, Wiedenbeck, Burnett, & Grigoreanu, 2010)

While the problem of how to allow end users to create software artifact attracts much attention, there are other issues related to different stages of software life cycle, such as design, testing, debugging. (Ko, et al., 2009)

According to the review of end user software engineering activities made in (Ko, et al., 2009), requirements and design are rarely independent activities. End users often perform evolutionary prototyping as a design, which finally leads them to the ready artifact(Cao, Riche, Wiedenbeck, Burnett, & Grigoreanu, 2010). End users prefer opportunistic design approach and do development without a plan how exactly they are going to achieve their goal. Mashup development is a typical example of the approach end users utilize for the design. (Hartmann, Doorley, & Klemmer, 2008)

Testing is evaluation of the how correct does the program work based on the program outputs. End users rarely perform systematic testing because that require some additional time to spend when the resulting application is ready, however they generally would like the Web applications to work correctly.

A few en user developers perform organized testing activities for their Web applications, however they report that they have some personal quality standards of the result and they do some checks. (Rosson, Ballin, & Nash, 2004) Testing activities are complicated for the end users. (Cao, Riche, Wiedenbeck, Burnett, & Grigoreanu, 2010)

Support of testing activities should be integrated with the process of creating the program. An approach called WYSIWYT (What You See is What You Test) is successfully used in spreadsheet testing and improves the effectiveness and efficiency of the testing procedure and suits end users who do not have any background knowledge in performing testing activities. (Rothermel, Cook, Burnett, Schonfeld, Green, & Rothermel, 2000) Using this approach, the code being tested is visible for the user and it is possible to incrementally check whether the part of the code works correctly. In spreadsheets user can see the immediate feedback from each cell and thus examine the correctness of the cell formula right away.

For end users debugging, the process of understanding how does the program behave, is also a challengeable task because of several reasons. End users have to understand the behavior of the program, which is written on the abstract level. Users do not see the link between the source code and the running application and thus it is difficult to them to find the place from where the error comes from. (Myers, Pane, & Ko, 2004)

In (Klann, Paterno, & Wulf, 2006) authors report that is very important to support the detecting and correcting accidental errors, which may be difficult to locate later and provide simulation environment, undo mechanism and history of changes.

2.5. Challenges and expectations of end user Web developers

This Section presents first part of the review of end users needs and problems. It investigates barriers and expectations of end users based on end user studies. User experience of using of existing tools developed in research and commercial sectors, which is a second part of the review, is presented in the Chapter 3.

There is a large body of research about needs and problems of professional Web developers as well as pure end users. However, there is no much analysis of middle category, end users who build Web applications and contribute different content. (Rosson, Ballin, & Rode, 2005)

The complexity and scope of Web applications may vary from static page to complex interactive enterprise application. However, end users are hardly going to implement distributed authoring system or online marketplace themselves. In most cases user tasks are simpler, e.g., personal Web page.

The results of the survey of informal Web developers, people who do some Web programming but do not have appropriate educational background (Rode, Rosson, & Quinones, 2006), showed that 30% of end user needs could be achieved with the application that provides basic data management functionality: collection, storage and retrieval. Another 40% of requirements can be fulfilled with set of general applications, such as resource reservations, shopping cart and payment, message board, content management system and calendar. According to another study (Rosson, Ballin, & Rode, 2005), end users create Web applications to support tasks achieved from their work, hobby, interest or community issues, school and family related occasions.

Ginige and Murugesan defined six categories of Web applications. (Ginige & Murugesan, 2002) Their work was refined and extended in (Ginige, De Silva, & Ginige, 2005), where authors specify the following categories: information; search, directory and dictionary lookup; e-commerce sites; entertainment, interaction and messaging; collaborative sites; Web development tools. In (Ginige, De Silva, & Ginige, 2005), the categories are analyzed against difficulties reported by informal Web developers.

In (Rosson, Ballin, & Nash, 2004) authors describe individual end user developers that represent the target group of end users with wide range of background tasks and skills. Later in their work they analyze how those end users develop Web applications and what problems do they face during the development. All interviewees used end user development tool to create their applications. They report that they faced the difficulty of converting the formats into each other, interacting and changing between different tools. End users also encounter problems while attempting to reuse poor code written by other people and integrate it into their application.

The study of typical high-level components, concepts and features of interactive Web application is described in (Rode & Rosson, 2003) based on the investigation of existing Web applications. Later, in (Rode, Rosson, & Perez-Quinones, 2004) those components are presented to end users, who have some knowledge of HTML and have used WYSIWUG editors, but have minimal or no background in programming. Authors analyze how end users think about the components and whether they utilize them in their applications. That reveals another barrier of end user Web development: how end users think about different issues typically addressed by programmers when developing Web application. It appeared that end users mental models are on a higher level of abstraction and they often do not have accurate understanding of technical implementation of typical functions, such as session management, authentication and authorization, database schemas, search process, etc.

The study about end users who use mashups is described in (Zang, Rosson, & Nasser, 2008). There are tools that aim to eliminate the difficulties of programming of Web mashups. However, users report that they face problems when they want to achieve more complex task using the tool. Finally, less experienced users cannot create a mashup. They also complain about lack of the documentation and examples.

Users reported they would like to have ready-made templates, wizards, layout assistance, direct manipulation and drag-n-drop. In (Rode, Rosson, & Quinones, 2006), authors claim that the tool that supports implementation of typical functionality of Web application on high level of abstraction can facilitate end users in the process of Web development. However, end users highlighted the importance of having full control of the low-level details.

According to survey described in (Rosson, Ballin, & Rode, 2005), end users tend to implement the same features in the applications they build as professional programmers, but they have less success. As for their expectations from the tools that support their work, end users asked for easy-to-use tool with support of WYSIWYG interface development, which can be easily integrated with other tools. End users report they prefer to use visual user interfaces instead of code-oriented. In (Rode, Rosson, & Quinones, 2006) authors report that most of end users asked for “Word for Web applications”, a tool that would support WYSIWYG paradigm, drag-n-drop, wizards, examples and templates, but yet allow developers to control the source code.

There are sophisticated end users, who have the same level of knowledge of the certain technology as professionals. Those end users are forced to work with the tools designed for professionals, however they face the gap between mental models and approaches end users and professionals tend to use, that was mentioned above. (Costabile, Mussio, Parasiliti Provenza, & Piccinno, 2008)

There is also a possibility that end users become more skilled while they are using the tool or the technology. Users report some of their skills are self-taught. (Zang, Rosson, & Nasser, 2008) They use instructions, tutorials, and community-related sources, such as discussion forums, sharing and reusing of tips and examples. (Rosson, Ballin, & Rode, 2005) End users say they often use code of other people as an example to learn. (Rosson, Ballin, & Nash, 2004)

Wide spread of the Internet and computers motivate end users to learn new technologies. Users report they prefer a tool that is flexible enough to support growing of their knowledge and needs. (Rode, Rosson, & Quinones, 2006)

Although the notion of end user is different from novice user, it may be useful to consider end users as learning novice users. (Myers, Ko, & Burnett, 2006) (Repenning & Ioannidou, 2006)

From this point of view it becomes important to consider learning barriers, described in (Ko, Myers, & Aung, 2004), while designing the tool.

2.6. Gentle slope of complexity

Even though there are end users, who are competent in the technology being used, lack of technological knowledge is native problem for end user development. Since research in Web engineering initially was more focused on professional programmers, the tools and technologies were designed for people with high knowledge of the area. (Rode & Rosson, 2003) Consider the situation, when unskilled end user attempts to develop a Web application from scratch. The user has to achieve sufficient level of knowledge of the whole set of underlying technologies, before the development can be started. Curve marked with ① at Figure 3 shows the steep incline that end user has to conquer, even if the functionality of the developing program is relatively simple. Moreover, user can make considerable effort to learn the technology, but still do not get any improvement in what he can achieve with the language. (MacLean, Carter, Lovstrand, & Moran, 1990) People, who only start programming, need to see that they really can do some progress and get valuable result. (Kelleher & Pausch, 2005)

End users often cannot overcome steep barrier of learning complicated technology that is used by professionals. If user meets insurmountable barrier, most probably it will stop the user from using the tool. (Ko, Myers, & Aung, 2004) Therefore, a tool for end users should keep the proportion between challenges caused by a development and knowledge required from a user and avoid high learning barriers. (Repenning & Ioannidou, 2006)

One solution to this problem can be a tool or technology with *gentle slope of complexity*, which is marked with ② at Figure 3. This concept emerged twenty years ago as a solution for problem of creating a system that can be successfully tailored by non-professional users with different level of skills. (MacLean, Carter, Lovstrand, & Moran, 1990) The idea behind the concept is to reduce the slope of the incline. So, the progress, which user makes, becomes more consistent with the effort spent. Users can increase the level of complexity from the one they are already familiar with to yet another one, which is more challengeable. Because of the gentle slope of the complexity curve, the upgrade requires relatively small increment in skill.

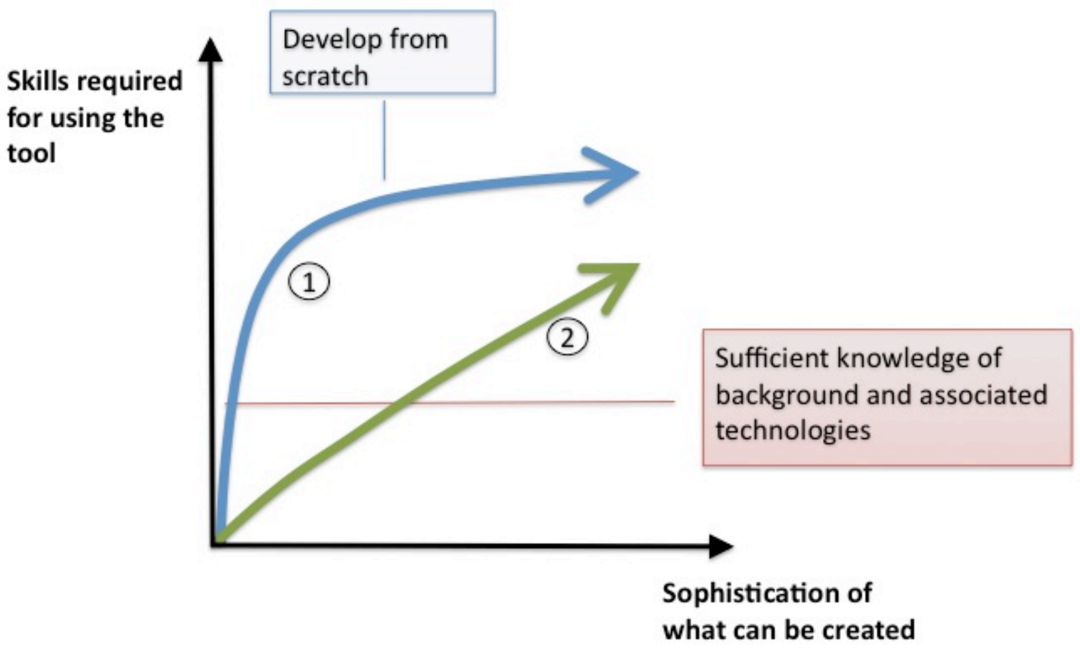


Figure 3 Actual and desired slopes of complexity for end user who develops a Web application from scratch

Support of the gentle curve is critical for the end user development. (Ko, et al., 2009) Nowadays, the approach is increasingly popular in end user development domain. Special tools aim to achieve gentle slope of complexity rise for different activities performed by end users. (Morch, Stevens, Won, Klann, Dittrich, & Wulf, 2004) (Rode, Bhardwaj, Perez-Quinones, Rosson, & Howarth, 2005) (Lieberman, Paterno, Klann, & Wulf, 2006)

In (Lieberman, Paterno, Klann, & Wulf, 2006), authors identify two levels of activities in end user development. First class contains activities that allow users to *customize* existing tools to their personal needs by selecting one of the available options. User can *tailor* the system by manual customization or use adaptive system that manages customization automatically depending on user's activities in the system. By definition, end user tailoring happens after initial design of the system made by professional programmers. (Won, Stiemerling, & Wulf, 2006) On this level users often get a predefined set of parameters they can change.

Second level contains activities of more advanced users that are *creation and/or modification of the software artifact*. Users can create programs from scratch or modify existing ones. End user development research domain offers several techniques that are often used to help end users to perform the activities, such as *programming by example, visual languages, direct manipulation, design at runtime,* that will be reviewed in detail in the Chapter 3.

The tool can support both types of activities and provide gentle rise of complexity level when moving from lower to higher stage. (Lieberman, Paterno, Klann, & Wulf, 2006) To achieve such flexibility the tool should provide a range of modification activities with different complexity levels, which are more powerful then simple parameterization but less complicated then programming.

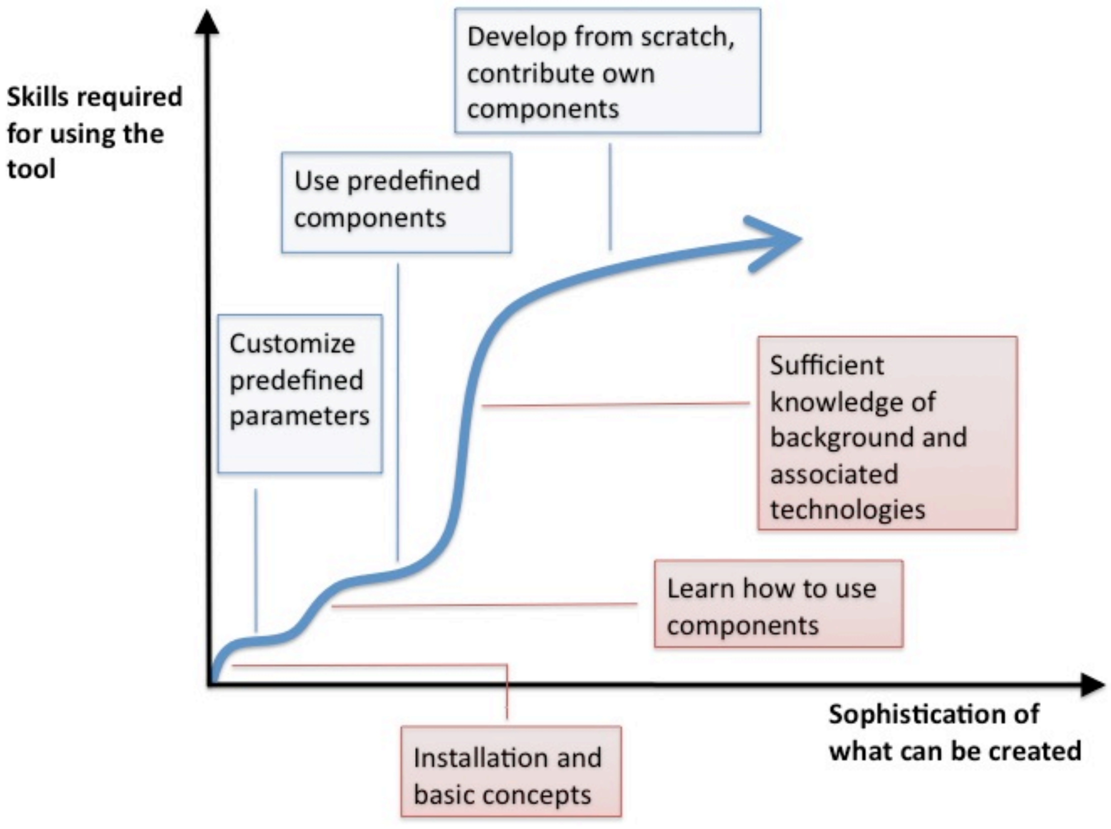

Skills required for using the tool
Develop from scratch, contribute own components
Use predefined components
Customize predefined parameters
Sufficient knowledge of background and associated technologies
Learn how to use components
Installation and basic concepts
Sophistication of what can be created

Figure 4 Slope of complexity for visual Web application development tool that allows customization and using of components

As an example, authors suggest to have a component-based system with 3 levels: on the first level user can set parameters; on the second level user is allowed to build the resulting application by combining existing components; on the last level user can program from scratch and contribute new components to the system. Possible complexity rise for this approach is displayed on the Figure 4.

In comparison to Figure 3 it has more gentle slope of complexity. Initial single cliff is now divided into several smaller cliffs. On the other hand, a user still has to overcome considerable barrier when learning how to contribute new components. In (Won, Stiemerling, & Wulf, 2006) authors propose that having additional intermediate levels of modification, such as possibility to view and modify source code of component, can help to achieve more gentle slope of complexity.

Each terrace on the depicted slope represents a group of people with some level of knowledge, habits in tool usage and tasks they achieve. If terraces are separated with steep inclines, it is relatively difficult for a person not only to upgrade to the next level but also to communicate with people from higher group. (MacLean, Carter, Lovstrand, & Moran, 1990)

3. Related work

This Chapter contains the review of existing solutions to the problems of end user development. It describes different approaches both from research domain and commercial sector and discusses their advantages and disadvantages. It also presents available Web development tools and describes user experience of using those.

3.1. Review of related research

The main goal of end user development is to allow end users to create software artifacts. Generally, the problem of programming is in translation thoughts of the users into programming language or any other form, which is interpreted by computer. According to extensive review of different tools and languages that attempt to make programming more accessible (Kelleher & Pausch, 2005), the main goal is to eliminate complicated mechanics of programming. Systems and languages that empower people to program aim to give end users the possibility to build as much as possible.

Two different approaches are used to overcome mechanical difficulties. First approach focuses on making programming languages more transparent and understandable for end users. Even though users are supported by programming environment, this approach assumes that the users write the code themselves. Second approach is to provide a tool, where end users can achieve the result without actual coding, e.g., by using visual programming, direct manipulation, and programming by example.

3.1.1. Languages

A lot of users are engaged in programming by providing more simple and understandable languages. (Scaffidi, Shaw, & Myers, 2005) In order to make the transition from ideas to programming language more smooth and less challengeable for end users, there are studies that aimed to construct more natural programming language. (Myers, Pane, & Ko, 2004) Using those languages, end users do not have to utilize complex mental models of professional languages, but rather express their ideas in the same way as they think about them. Using of natural languages has positive effect on debugging activities. There are studies about mental models of end users and how do they think about programming. (Rode, Rosson, & Perez-Quinones, 2004) (Myers, Pane, & Ko, 2004)

There are studies about creating the programming language that is similar to the English language. Users are allowed to write textual specifications; the development environment interprets and executes those specifications.

Another close branch is providing *domain specific* or *task-specific* languages, which terms and paradigms are familiar for concrete target user group or concrete task and thus are more understandable for end users. (Lieberman, Paterno, Klann, & Wulf, 2006) In 1993 Nardi described the advantages and disadvantages of utilizing task-specific language approach and those disadvantages are still relevant now. (Nardi, 1993) First, users have to learn new language to start the programming and they have to switch to a new system when they have a task from another domain. Second, each task domain should have its own tool.

Nardi claims that interaction techniques along cannot provide end users with an ability to create applications by themselves; he argues that interaction techniques should be combined with task-specific languages. (Nardi, 1993) This approach is employed in different tools and researches.

Repenning and Ioannidou define the notion of *meta-domain oriented language,* which does not use so specific terms as as a domain-specific language, but still less general than programming language. (Repenning & Ioannidou, 2006) According to authors, spreadsheets, which are used by people from different domains for different tasks, are an example of meta-domain oriented language.

Another example is a Curl language, described in (Hostetter, Kranz, Seed, Terman, & Ward, 1997). It addresses the problem of combining different technologies in order to create an interactive Web application. The language syntax supports smooth move from writing simple formatted text, based on HTML tags, to powerful object-oriented programming by combining those. Users can select the complexity level that is appropriate to their tasks and knowledge, and that makes the language accessible to authors with different skill level, ranging from novices to professional programmers.

3.1.2. Programming environments

On the border between visual tools and simplified languages there is a group of programming environments for end users, which are widely reviewed in (Kelleher & Pausch, 2005) A special programming environment addressed to end users can facilitate the development and hide programming difficulties. (Pane & Myers, 1996)

Programming environments aim to help users in learning of the programming language and eliminate programming difficulties. (Myers, Ko, & Burnett, 2006) A major task for programming environment is to reduce the number of syntax errors. (Repenning & Ioannidou, 2006) For example, a syntax-directed tool called Alice forces users to program using syntax templates and allows them only to fill the gaps with necessary information. (Myers, Pane, & Ko, 2004) This helps to avoid errors and introduce the language to the user. Other typical features that are provided are context error correction suggestions, templates, integrated help, etc. (Kelleher & Pausch, 2005) Inherited from professional development environments, highlighting of reserved words and symbols helps to notice syntax errors. (Repenning & Ioannidou, 2006) This technique can be expanded to highlight the structure of the program using different color schemes.

3.1.3. Reuse practices and sharing

As it was discussed in Section 2.5, end users often utilize reuse practices when performing development activities. In (Ko, et al., 2009) authors define two types of reuse: *composition,* that is combining and gluing predefined components (Hartmann, Doorley, & Klemmer, 2008), and *modification,* that is using and modifying existing code in order to adapt it to the new context.

Reuse is a known and popular practice in professional programming. It reduces the time spent on the development and number of errors by using the code that is already developed and tested. Composition increases maintainability and improves the readability of the program because becomes more structured. Reuse practices bring the same advantages to end user development. However, searching, actual reusing and sharing the code in end user development are more focused on rapid achievement of the resulting application. (Ko, et al., 2009)

On the other hand, in end user development, reuse often makes the goal of creating the application feasible, because users are not able to implement the desired functionality manually. (Ko, et al., 2009) End users effort is shifted from actual programming to searching for appropriate reuse abstractions and using them in their program. (Hartmann, Doorley, & Klemmer, 2008)

Facilitating activities of searching, reusing and sharing abstractions is the main goal of the environment that supports reuse in end user development. First activity is to find the code or component to reuse. (Gaedke & Rehse, 2000) From end user point of view, this process starts with general understanding that there can be a code for reuse that can help to implement the program. After that end user has to imagine what functionality is needed, look for appropriate candidate, decide whether this candidate suits the task and finally use it in the program. An effective search engine provides a possibility to search using different options, including rating and tags (Grammel & Storey, 2008); each reusable element has extensive description; it is relatively easy to try the component or code and see how it works to make the decision. There are search tools that advise or filter reusable abstractions based on monitoring users activity or analyzing users code. (Ko, et al., 2009) The compatibility of the component can be also analyzed by the system in order to prevent user from errors. (Won, Stiemerling, & Wulf, 2006)

When the component or code example is found, it should be reused in the program. According to (Ko, Myers, & Aung, 2004), the problem may occur if the procedure of using the component or code is not obvious for the users and there is lack of documentation about the procedure. Another challenge for a user, which appears when a component is built into the program, is to understand the link between the reusable abstraction and the final output. (Ko, et al., 2009)

The last aspect of reuse is sharing of the reusable code. Cooperation is becoming highly important issue of end user development. With increasing popularity of social communication, a successful tool should provide possibility to share and reuse solutions and discuss problems (Klann, Paterno, & Wulf, 2006) End users often reuse their own solutions. (Rode, Rosson, & Quinones, 2006) A lot of environments allow end user to contribute reusable abstractions. (Hartmann, Wu, Collins, & Klemmer, 2007) (Grammel & Storey, 2008) In order to support end user sharing activities, the procedure of creating the abstraction should be relatively simple. However, the difficulty and the cost of maintaining the database of reusable artifacts increase if end users are allowed to contribute content there. The environment should provide advanced search, filter and Web 2.0 features such as commenting and rating. (Ko, et al., 2009)

Component-based approach

The idea of composite reuse for end user Web development has evolved from component-based development. Component-oriented method, inherited from software engineering, is utilized in professional Web development as a Component-Based Web Engineering. (Gaedke & Graef, 2001) For example, WebComposition Process Model and WebComposition Markup Language (WCML) aim to bridge the gap between coarse-grain design concepts and fine-grained implementation details and also help to handle variety of technologies used for Web development. (Gaedke, Segor, & Gellersen, 2000)(Gaedke & Rehse, 2000) The approach allows defining components and creating Web applications as a composition of these components. Component is a module that encapsulates some functionality; it is independent and substitutable. The functionality that is represented by component may vary from atomic interface block to complex navigation feature; it can be composed from other previously created components. The WCML language, which is an extension of XML, allows to define abstract design concepts and low level implementation details of the component and also allows to store meta information used for search purpose. Authors also investigate how to facilitate compositional reuse by providing a repository for shared components and improving searching activities. The proposed repository provides meta-information about the components, representations of the component and support different classification methods in order to facilitate developer to find the component that suits the problem.

Another application of compositional reuse is creation of composite Web services. In (Yang & Papazoglou, 2006) authors propose to employ reuse by combining existing services to create a new valuable service. They investigate different types of composition logic, describe the XML-based language that is used to define Web component and to specify how those components are combined and executed, and finally propose a framework that supports entire life cycle of the service composition. Same approach is employed in end user mashup editors that are focused on mashing up data. (Grammel & Storey, 2008) (Yu, Benatallah, Casati, & Daniel, 2008)

Composite reuse is widely applied in end user development. It allows end users, who apply the component in their program on a high level, not to look into technical details. (Won, Stiemerling, & Wulf, 2006) (Ko, et al., 2009)

Component-based approach is used in end user tailoring that is adjusting the system to the needs of the end user after professional developer has designed it. (Ko, et al., 2009) For example, Won, Stiemerling and Wulf give an overview of tailorable systems and describe their idea of applying component-based approach in order to achieve more tailorable system for end users. (Won, Stiemerling, & Wulf, 2006) Composition of components provides middle level of complexity, which acts as an intermediate stage between parameterization and programming. Components themselves can consist of more low-level components. However, their approach does not support the possibility to modify the component.

A challenge of composite reuse is to design a sufficient set of components. When designing an abstraction for end users, it is important to select the concept that will be represented as a component and choose the level of abstraction of the concept so that component functionality corresponds user expectations. That facilitates end users to use the abstraction and generally allow them to achieve more advanced goal. (Ko, et al., 2009)

Besides analysis of needs of end users, one way can be investigation of typical building blocks of existing Web applications. (Ginige, De Silva, & Ginige, 2005) Another option is involving more sophisticated end users in the process of creating component database. It helps to provide components that suit end user tasks and thus provide components that are need to users. (Gaedke & Rehse, 2000) However, that requires more efforts for maintaining the database and analyzing the quality of components.

If reusable component does not match the task, it can be tailored by the user. Tailoring, e.g., using parameters, can change the behavior or appearance of the component, but does not include modification of the component source. However, there are end user development systems that provide set of customizable components and allow to modify the components source code if the desired functionality cannot be achieved by simple customization. (Ko, et al., 2009)

Implementation of component-based approach can have different levels of visibility of component source code, starting from black box components, where the implementation is hidden, to white box, where the source code is visible to a user. (Won, Stiemerling, & Wulf, 2006) End users expect to have a full control over the system and thus need to have an access to component source and ability to modify it. (Cao, Riche, Wiedenbeck, Burnett, & Grigoreanu, 2010) (Hartmann, Wu, Collins, & Klemmer, 2007) It is important to make components be easy to use; the procedure should not require low level technical details, otherwise it eliminates all advantages of component usage for end users.

Some component-based systems have a focus on providing a fixed set of components, assuming that they cover all possible user needs, and do not consider the problem of creating new component by end users. (Myers, Hudson, & Pausch, 2000) In professional programming components often cannot be modified or extended, because they represent an external library or API. (Ko, et al., 2009) In that case the mechanism of creating new component can be difficult and thus inappropriate for end users. Procedure of sharing should be transparent and do not require to have additional knowledge of system architecture or special notations. In addition, the structure of components and method of their use and extension needs be transparent and meaningful to the user, so user can design and perform changes to the components. (Lieberman, Paterno, Klann, & Wulf, 2006)

Component-based approach is used to achieve gentle slope of complexity rise. (Lieberman, Paterno, Klann, & Wulf, 2006) In (Ginige, De Silva, & Ginige, 2005) authors describe how they applied component-based approach in order to create a tool for Web application development in the domain of small and medium enterprise Web applications. The focus of the tool is to facilitate development of applications by domain expert end users.

In (Rode, Bhardwaj, Perez-Quinones, Rosson, & Howarth, 2005) authors describe their application of component-based approach for achieving gentle level of complexity in the tool for end user development of data collection and management Web applications. The tool provides several layers of modification complexity, starting from customizing templates to modifying and extending the component framework and editing PHP code.

(Fiala, Hinz, Meissner, & Wehner, 2003) describe the use of component-based approach in creating adaptive personalized Web applications. They provide high level declarative XML based components of different abstraction levels to support Web authoring and generation.

Utilizing examples

Utilizing examples and copy-paste are common practices used in opportunistic style of design and programming.(Brandt, Guo, Lewenstein, & Klemmer, 2008), (Kim, Bergman, Lau, & Notkin, 2004) (Hartmann, Doorley, & Klemmer, 2008) Results of users survey have shown that users often leverage existing examples to create the application and consider using of example approach rather useful.(Rode, Rosson, & Quinones, 2006) (Zang, Rosson, & Nasser, 2008)

There are different patterns of using the examples. *Programming by example modification* was noted to be useful for end user development area. (Nardi, 1993) Users can utilize a source code written by them or shared by other people by copying and modifying it in the application being developed. (Zang, Rosson, & Nasser, 2008) Using this approach, users can achieve more complex resulting application, however they face problem of fixing errors possibly hidden in the copied code snippet. (Rosson, Ballin, & Nash, 2004)

The approach used in tools for mashup creation is to copy the whole Web application developed and shared by other person and modify it to achieve desired functionality. (Grammel & Storey, 2008)

Other approach widely utilized by end users is using examples of what they want to program to gain some knowledge of the problem and later implement the solution themselves. In this case example act as a help artifact, which demonstrates how to solve a problem. Survey of (Rosson, Ballin, & Nash, 2004) shows that end users utilize other's code as example for learning rather then use it for copy-pasting into their applications. Some users claim that they prefer to gain some knowledge from the example and then implement the functionality themselves. Reuse can be a source of errors, if external component or snippet does not work as user supposed.

3.1.4. Visual approaches

This section describes essential solutions designed to eliminate the difficulty of programming by allowing end users to create an application without actual programming. (Kelleher & Pausch, 2005) The following approaches are discussed: *visual programming, programming by demonstration, programming at runtime*.

Visual programming

Using of v*isual programming language* allows users to create a program by manipulating graphical components and connectors rather then by writing the code. Elements of the program are represented in graphical objects. User adds, customizes and connects the objects visually in order to define the functionality of the program. This approach is often employed in data-flow programming, where the program defines how the data is processed. In this case each block represents the source of data or action to perform and connectors define the flow. Several data mashup tools are designed based on this approach. Visual programming reduces the effort spend on learning language syntax and helps to eliminate syntax errors. (Repenning

& Ioannidou, 2006) There is a tendency to consider visual programming more easy to learn by non-professionals because of the graphical representation of the concepts and structure, however studies showed that it has some disadvantages that need to be taken into account. (Myers, Ko, & Burnett, 2006) Two most important disadvantages are inefficient use of screen space and high viscosity of the visual program representation. The efficiency of visual representation depends on how does this representation fits real end users, their background and tasks. (Repenning & Ioannidou, 2006)

Programming by demonstration

Programming by example approach allows to create a working program by demonstrating of what is the result of the program work. User provides a set of examples and after that the system generalizes them and creates the program logic.(Lieberman, 2001) Several systems employ this approach (Grammel & Storey, 2008). Because the Web consists of example applications, the approach can be successfully used for Web development. (Ko, et al., 2009) Users browse the existing Web applications and find examples that they need in their applications. After that they use the tool to copy either the entire content or the structure of interface widgets to their application. (Hartmann, Wu, Collins, & Klemmer, 2007)

Programming by demonstration approach is successfully used in creating mashups for collecting the data from existing Web applications and processing it. (Grammel & Storey, 2008)

For example, tool called d.mix combines programming by example and programming by modification approaches in order to lower the threshold for creating Web mashup by end users. (Hartmann, Wu, Collins, & Klemmer, 2007) Using the tool user can browse the annotated Web sites and select examples of desired functionality and reproduce them in the developing application by means of *parametric copying*. The tool engine processes the examples and detects the underlying logic to integrate it into the application. Users can lately view and modify the code. Empirical study described in the paper shows that users find the approach useful for rapid creation of the applications and making the required knowledge threshold lower.

The concept of *incremental development* is similar to the concept of gentle slope of complexity, described above. It assumes that end user can add the complexity to the program incrementally, without having steep cliffs in skills required from user. Other key characteristic of a tool that supports incremental development approach is that it is possible to run the application when the development is not finished or test part of the application. (Repenning & Ioannidou, 2006)

Programming at runtime

Trying to support active nature of end users, their intent to reach the final result and opportunistic approach to programming, Rode and Rosson proposed new paradigm for end user Web development called *programming at runtime*. (Rode & Rosson, 2003) The core idea of programming at runtime extends WYSIWYG approach and allows a user to *develop* and *use* the application simultaneously, without any compiling activities or time-consuming switching between designing and running modes. In one single view of the prototype tool the resulting application itself is shown and user can directly edit it by changing parameters or adding and managing using direct manipulation. On the contrary to preview mode that is used in other tools, the functionality of application is not restricted, so it is possible to really use the application. According to authors, the main advantage of the approach is that user has immediate feedback and sees the link between the process of building the application and the result. The reliability of the resulting application can be improved because the live testing is built into development process.

Same approach is utilized in (Hundhausen, Farley, & Brown, 2009) to support visualization of programming process. Prototype tool, described in the paper, provides a view where user can modify the application via direct manipulation and see the immediate feedback.

Other authors, e.g., (Cao, Riche, Wiedenbeck, Burnett, & Grigoreanu, 2010), also mention that for successful end user tool it is important to have appropriate cost of running or testing of the application being developed. If the cost is too high, end user can fail to achieve the result or miss the connection between source code or design and the result.

(Lieberman, Paterno, Klann, & Wulf, 2006) also claim that one of the key features is the possibility to perform changes in the developing application while it is running without spending time on compilation.

3.1.5. Interaction styles

Direct manipulation

Shneidermann introduced *direct manipulation* term in 1983 (Shneiderman, 1983), and since that time it has become widely used in many domains of computer science. In a contrast to manual editing of the source code and performing script commands in the command line interfaces, direct manipulation approach provides visual representation of the information and allows user to manipulate it. He describes three main principles of direct manipulation: objects of interest are visible; the interface supports rapid, incremental activities for changing; instead of modifying the source code, changes are made by manipulating the objects based on actions from physical world. (Shneiderman & Plaisant, 2010) According to Shneiderman, it is not applicable for some tasks, but generally it helps end users in learning basic functionality of the tool and using it without syntax errors.

Obvious example of direct manipulation is an activity of deleting a file in the operating system by dragging a file icon to the trash bin. Both file and trash bin are represented with icons with metaphor pictures. Another well-known example of direct manipulation is WYSIWYG word processors. Using of WYSIWYG it can brings two obvious advantages: a user can see how the developing document will look like and it is not required to learn any commands or controls to manipulate the document. There are many more examples of direct manipulation, starting from spreadsheets and operating systems and continuing with video games and augmented reality. (Shneiderman & Plaisant, 2010)

A successful direct manipulation application provides representation of the reality. In some cases it may be difficult to switch to direct manipulation from direct coding, but nowadays one can hardly imagine why it is necessary to learn complex syntax while it is possible to manage things visually. (Shneiderman & Plaisant, 2010)

In the end user domain direct manipulation approach is employed to help end users to understand and intuitively manipulate information structures. Drag-n-drop of the components of the application, WYSIWYG editors, visual representation of concepts are typical examples of direct manipulation and they are successfully used in end user development tools. (Pane & Myers, 1996) (Rode, Rosson, & Perez-Quinones, 2004) (Grammel & Storey, 2008)(Kelleher & Pausch, 2005)

In (Hundhausen, Farley, & Brown, 2009) authors describe their investigation about combination of direct manipulation and manual text edition. The tool, created in scope of this work, provides two windows where user can modify the application both by editing the text and performing direct manipulation activities with visual objects. An experimental study shows this approach helps novice end users to obtain notably better outcome. Moreover, it shows that this approach facilitates learning of the conventional programming language and provides smooth transition from direct manipulation to text edition.

User interface metaphors

User interface (UI) metaphor is an approach that is related to direct manipulation. The idea behind the approach is to simplify complex user interface by grouping UI elements, information and related tasks and representing them as a set of concepts, which are already familiar for the user. (Neale & Carroll, 1997) Metaphors can be taken both from physical world and from other existing known systems.

Metaphor is a way how a designer can explain one thing through another. Practically it works in a way that users do not know how to achieve the task in new system, but they recognize the metaphor being used and it gives a hint how to proceed with the task.

In order to make metaphor work, two main conditions should be met: metaphor should be well known for the target user group and it should fit the task it represents. (Neale & Carroll, 1997) Using unknown metaphor obviously makes user interface more complex. If metaphor gives wrong impression or turn user into wrong direction, then it even makes UI more error-prone. Otherwise, metaphor can bring many benefits to user interface if properly selected. (Barr, Biddle, & Noble, 2002)

Initially using of metaphors in UI was designed to improve learnability, however now they are also used to facilitate usability. (Neale & Carroll, 1997)

Other interaction styles

Besides direct-manipulation, there are other strategies to be used in user interface, such as menus, forms and dialogs. If competently designed, these approaches are effective to achieve tasks, which cannot be solved with direct manipulation. (Shneiderman & Plaisant, 2010)

Menus are effective when there is a set of items to choose from. Menus and forms should be properly organized; lack in organization leads to misunderstanding.

When it is required to enter many fields of data, form fill-in interaction style becomes appropriate solution. Depending on the task to solve, there are various types of form: from simple ones where user can enter text and press submit button to combination of fields, different interactive widgets and popups. When the form is designed, special attention is paid to the wording of field label texts and structure of the form. Field title and/or description formulated ambiguously or unclear can confuse the user or become a reason for error. Unstructured form, where fields related to different areas are mixed, looks complicated and can mislead the user. Also, validation of the user-entered data and fast feedback about errors are highly recommended. (Shneiderman & Plaisant, 2010)

3.2. Review of existing tools

In this Section, different existing end user tools for creating Web applications are reviewed. The purpose of this review is to investigate what main paradigms and interaction models are used in the existing systems and also what the main problems are. Good system design can hardly be created without knowing common solutions and problems in the area. (Sano, 1996) (Myers, Hudson, & Pausch, 2000)

According to the scope of work defined in Section 1.2, the review is focused on Web-based tools, which try to simplify Web application development for end users. The review is focused on different concepts implemented in existing tools, not on technologies for Web development.

During the review each tool will be checked against the following criteria, which are important to the idea of XIDE (based on review made in (Fraternali, 1999)):

- Reuse (Does the tool allows to reuse something? What are reusable objects? Is it possible to create new reusable objects?)
- UI paradigm (What is main UI paradigm used? Is it useful? What design decisions are used?)

Different reviews and classifications of Web development tools were investigated to analyze what general types of tools exist and what application types fit into the scope of this review. (Fraternali, 1999) (Jazayeri, 2007) Finally, the following types are included: Web-based HTML editors, form builders, Web mashup editors, and content-management systems.

3.2.1. Web based HTML editors

HTML editors natively use WYSIWYG paradigm to offer non-technical users a way to create Web applications without any programming. (Jazayeri, 2007) When creation of Web applications by non-experts became popular, HTML editors were the first appropriate tools for end user Web development. (Rode, Rosson, & Perez-Quinones, 2004) Currently, HTML editors offer fancy WYSIWYG editors, media widgets and support of managing and publishing of the created applications. One example of this kind of tools is Google Sites[1]. It allows to add media widgets by selecting them from the list and then manage them by drag-n-drop. HTML editors are focused on static pages for information representation, not on interactive Web sites.

3.2.2. Form builders

Form builders offer user a way for designing Web form that can be later published and other users can access and fill it. Resulting Web page functionality is restricted by a concept of Web form, what means it can contain different types of fields to fill and submit button. Form builders are targeted to non-expert users and hence try to simplify form creating process as much as possible. For example, JotForm[2] offers drag-n-drop interface for adding and managing different UI elements in the WYSIWYG-like editor. The editor sketches structure of the form, however it does not show the form exactly how it will finally look like and does not allow to edit the source code of the form directly. Special preview mode is used to view the final appearance of the form. It also provides several tabs, where different features are grouped, so a user can configure the settings of UI element and the whole form. When the development has finished, user can publish the form on the server managed by JotForm.

Another example is Form Builder by Orbeon[3], which allows building XForms based forms. Generally, it has the same functionality as JotForm. The major difference is that FormBuilder gives advanced users the possibility to view and edit source code of the form.

3.2.3. Web mashup editors

Another type of Web applications is mashup. Web mashup is a Web application that combines of different data sources, services and Web APIs. (Yu, Benatallah, Casati, & Daniel, 2008) End users create mashups to address problems received from their daily life. (Maximilien, Ranabahu, & Gomadam, 2008) There are plenty of existing mashups and APIs to be used in mashup creation. By December 2010 there are 5300 mashups and 2400 API (ProgrammableWeb - Mashups, APIs, and the Web as Platform, 2010)

[1] Google Sites, https://sites.google.com/
[2] JotForm, http://www.jotform.com
[3] Orbeon Form Builder, http://www.orbeon.com/forms/orbeon-form-builder

Originally, creating a mashup is a difficult task, which may require knowledge of text parsing, pattern matching, databases, Web services APIs, technologies to create both client-side representation of information and server-side information processing. There are different languages and frameworks for advanced users, who have appropriate programming skills, for example IBM Sharable Code[4], Google Mashup Editor[5], etc. (Beletski, 2008) (Maximilien, Ranabahu, & Gomadam, 2008) These tools are out of the scope of this review.

Nowadays, there are a lot of mashup environments also for non-professional users. Well-known examples are Yahoo!Pipes[6], Microsoft Popfly[7], QEDWiki[8], Intel MashMaker[9] and many more. (Yu, Benatallah, Casati, & Daniel, 2008) (Grammel & Storey, 2008) (Maximilien, Ranabahu, & Gomadam, 2008)

However, there is a tendency to discontinue some mashup tools, which were popular several years ago, e.g., Google Mashup Editor in 2009, Microsoft Popfly in 2009, QEDWiki is replaced by IBM Lotus Mashups, however the approach remains the same. On the other hand, new mashup editors appeared and new mashups are created everyday, what means that the area is popular and users are interested in creating this kind of applications (Yu, Benatallah, Casati, & Daniel, 2008) (Ennals & Gay, 2007)

Although mashups are more focused on combining services, mashup tools are relevant for this review. Mashup editors are considered as end user development tools and have a main focus on allowing non-professional user to create a working application by combining visual objects without programming.

Mashups tools natively have reuse-based approach for development, since each mashup combines existing services and content.

Most of mashup tools provide means for deploying the mashup on the server and maintain it, so user does not have to worry about running a Web server. Some of tools are implemented using technologies, which require additional plug-in installation in order to be able to use the tool. (Grammel & Storey, 2008)

---

[4] IBM Sharable Code http://services.alphaworks.ibm.com/isc/
[5] Google Mashup Editor http://code.google.com/gme/
[6] Yahoo!Pipes http://pipes.yahoo.com/pipes/
[7] Microsoft Popfly http://popflyteam.spaces.live.com/ (discontinued)
[8] IBM QedWiki http://services.alphaworks.ibm.com/graduated/qedwiki.html (discontinued)
[9] Intel MashMaker http://mashmaker.intel.com/web/

For example, YahooPipes mashup editor is a visual editor with several tabs: library tab provides library of modules that can be utilized during the development; canvas tab, where the mashup is designed; debug tab, where the resulting mashup is shown. Users are required to have basic programming skills and understanding of data structures and dataflow model. To create a data mashup, user selects modules from the library, drag-n-drops them to the canvas, wires them with the mouse, configures modules settings to make them fit the task and publishes the pipe.

Microsoft Popfly generally uses the same paradigm. One notable design solution is that user can switch between two zoom levels when creating a mashup. When configuring the component, user can zoom in to see only necessary component. When editing has finished, user can zoom out to see the overall structure of the system. Component models are extensible.

Intel Mash Maker is a browser extension, which allows end users to create mashups. It provides a library of widgets, which are rated by end users. A separate plugin must be installed to the browser before the tool can be used.

QEDWiki also provides predefined widgets to create a mashup. It has several views to perform different actions in order not to overload the interface. When it comes to page creation, user has to define a page layout first. After that, a canvas for widgets is generated. User can drag-n-drop widgets into places, defined by page layout. Widgets can be with or without user interface, depending on functionality. Widgets can automatically interconnect with each other to work together.

3.2.4. Content management systems and modern blog management systems

Initially content-management systems (CMS) and blog environment were focused on content presentation and blogging(Douglass, Little, & Smith, 2005). Nowadays they are developing to provide wider modern functionality, such as managing multiple users, events, error handling, and is said to allow users to create general-purpose Web application. However, users are required to perform additional low-level actions to achieve the functionality that is beyond the initial scope of the tool, e.g., a simple static page.

According to reviews of the market distribution, three systems dominate the market: Joomla[10], Drupal[11] and WordPress[12]. Depending on the point of evaluation (e.g., average download rate, current usage, brand strength measurement, etc.) the leader product can be different. (Open Source CMS Market Share Report 2009, 2009; Browser market share, 2010)

All three systems use similar UI paradigm and have similar basic workflow. This review if focused on the Drupal tool, because it has usability research results published. (Scollan, Byrnes, Nagle, Coyle, York, & Ingram, 2008)

Drupal user interface is based on the menus and pages where user can configure the settings. The content has to be created and managed using lists and wizard-based forms. Users complained about the lack of intuitive user interface. (Scollan, Byrnes, Nagle, Coyle, York, & Ingram, 2008) The tool has terminology metaphors (pages, stories, blocks, panels), which are widely used during the site development. However, they are not obvious to user since there is no graphical representation of these notions and their connections.(Scollan, Byrnes, Nagle, Coyle, York, & Ingram, 2008)

WordPress is quite similar to Drupal, despite the fact that it has simpler user interface and the functionality it provides is restricted to create a blog Web page. Although the latest version of WordPress allows some drag-n-drop, direct manipulation is not presented widely.

All three systems (Joomla, Drupal and WordPress) are extendable, however the developer is required to have high level of technology knowledge to contribute new functionality to the system. It is a common practice that developer community, not end users, implements these new items.

A notable disadvantage of those tools is that they require installation on user computer. WordPress has an online interface, however it provides limited functionality.

3.3. Conclusions

The review of existing tools shows there are a lot of tools and approaches, however none is all-purpose. Each tool has a target type of Web applications that can be created.

[10] Joomla! http://www.joomla.org/
[11] Drupal http://drupal.org/
[12] WordPress http://wordpress.org/

Different user interface paradigms are the most important aspect of this review. Most of mashup tools use drag-n-drop paradigm for adding and managing components. However, there is usually a combination of drag-n-drop, form filling and text editing. WYSIWYG paradigm is highly used for creating simple HTML Web pages. However, some form builders work so that a user selects the component from the library and then specifies the place it should be placed with mouse without actual drag-n-drop. Content-management systems usually support old style of interaction, where new components can be added by using standard UI elements like input fields, check boxes and buttons.

It is important to note that many tools use special mental metaphors: either for organizing the internal system data, like content-management systems or for representing system parts. Metaphors are widely used in designing of user interfaces, and it is reasonable to use them to simplify the interface and help user to learn it faster.(Shneiderman & Plaisant, 2010) It is important to check that selected metaphor increases interface usability, since sometimes using of metaphor confuses users.

Some of the tools offer a possibility to go beyond visual editing and view and edit source code of the application directly. This feature can be reasonable for advanced users.

It is common for reviewed tools to provide a hosting for created applications. However, some content-management systems assume that user will maintain the application manually.

Generally, reuse paradigm is represented in most of the tools; however the style of reuse differs depending on the type of the tool. Yu et al. proposes to consider mashup editors as tools for components (or widgets) integration based on some composition logic. (Yu, Benatallah, Casati, & Daniel, 2008) This classification can be extended over all tools in this review. According to that, the type of a component can be different, depending on how it behaves: *data component* only provides data for the application; *application logic components* are used to construct the application functionality; *user interface components* are used to construct user interface.

Mashups editors often have only data and application logic components, while HTML editors and content-management systems usually have UI components and sometimes application logic components.

One important aspect of component reuse is whether system allows creating of new components or extending existing ones. Content-management systems often provide such opportunity, as well as some mashup editors. However, the way in which the system can be expanded with the new component is usually not suitable for non-professional users, so extensions are made by members of developer community.

4. XForms and XFormsDB

This Chapter describes general information about XForms and XFormsDB technologies. After that, there is a Section that describes structure of XFormsDB application and its components. Difficulties and problems user can face during XFormsDB application development are described at the last Section.

As it was discussed in the introduction Section, XFormsDB framework was selected to be a core technology of creating applications in XIDE. XFormsDB is an extension of XForms, so XForms will be briefly described first.

4.1. XForms

XForms is the next generation of HTML forms in a sense that both are extensions of XML and used for creating Web forms. (Boyer, 2009) However, XForms overcomes several limitations of HTML forms, such as dependency on scripting languages and lack of tools to operate with the data on the client. (Dubinko, 2003) XForms introduces set of elements that simplifies Web form creation.

Unlike HTML, XForms supports Model-View-Controller (MVC) design pattern, whose key idea is to separate application's data from logic and presentation. (Gamma, Helm, Johnson, & Vlissides, 1995) This approach is widely used in Web development and is useful for interactive system creation. (Leff & Rayfield, 2001)

Unlike HTML, XForms supports asynchronous data submissions to the server. Because of this, user can continue interaction with XForms-based Web page while the submission is processed on the server.

Besides features described above, XForms has many other benefits, which facilitate creation of more interactive Web forms then using plain HTML. (Dubinko, 2003)

In addition, XForms can be used for creating more general applications than simple form. (Dubinko, 2003) XForms language is more usable for highly interactive Web application development in comparison to other declarative languages. (Pohja, Honkala, Penttinen, Vuorimaa, & Ervamaa, 2007)

But XForms-based application development still requires server-side coding in order to support communication with the database. (McCreary, 2007) XForms applications lack for high-level user interface features, such as user authentication and access control, error handling, and passing information between two XForms-based documents. (Laine, 2010)

4.2. XFormsDB

XFormsDB extends XForms in order to overcome the problems described above.

With XFormsDB it is possible to build interactive Web application by using only declarative languages and writing only client-side code. It is designed to simplify Web development for professional developers and even non-programmers. (Laine, 2010) Main features of XFormsDB are data management, both with and without synchronization, error handling, session management and access control.

XFormsDB utilizes MVC pattern as well as XForms. Model part of MVC is represented by instance data objects. Instance data defines what data will be used on the page. Each instance data is an XML skeleton, which specifies the structure of the information. Additionally it can contain initial data values. View part of the MVC pattern is represented with user interface controls, which are built in XHTML and bound to instance data. Controller part provides means for data management and communication with the database. It consists of XPath and XQuery expressions and submissions. (Malhotra, Melton, & Walsh, 2010) XPath is used to access different parts of the instances. XQuery is used to query collections of XML data from the database. Submissions are used to store data in the database.

4.2.1. XFormsDB application

According to (Laine, 2010), authoring XFormsDB Web pages involves using different declarative technologies for different purposes (see Figure 5):

• XHTML for document structure

• XFormsDB for data access and common server-side tasks and XForms for user interaction

• XML for data modeling and interchange

• CSS for visual layout and presentation

• XQuery and XPath for querying data

Additionally, different external Resources, e.g., images or JavaScript, that can be built into XHTML.

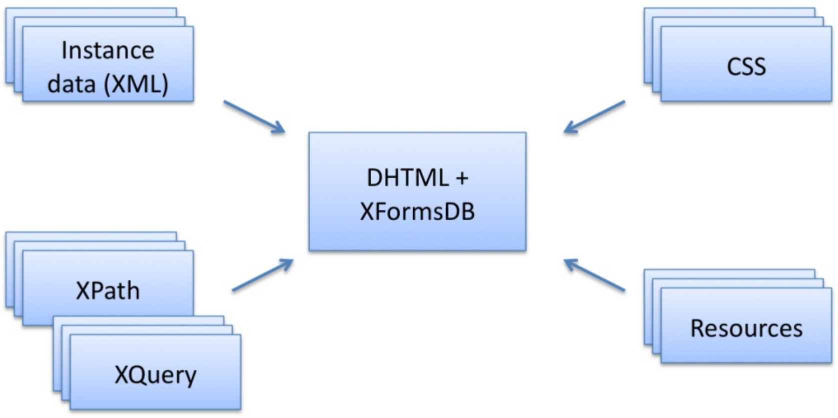


Figure 5 XFormsDB Web page components

On practice, each Web page has a source file, which defines page structure and user interface. Additionally, different components can be introduced, e.g., instance data, XPath and XQuery equations, CSS, external resources. Generally, each component can be either described in external file or specified directly in the source code. Using external files provides reusability and reduces complexity of the page source code. However, in case of a complex application, it can become difficult to organize all application files.

Except external component files, XFormsDB application also contains auxiliary files and folders used when the application is running. Several obligatory configuration files should be set up. They cover different technical issues, for example, how the application should be deployed on the server or how to connect to the database.

In order to run XFormsDB application, there should be a Web server with servlet container and Exist database.

4.3. Challenges of development using XFormsDB framework

Although XFormsDB was designed in order to simplify Web development, end users face several challenges during development from scratch using pure XFormsDB technology. XFormsDB is relatively new technology and there are no published papers about end user experience of using it yet. Thus the challenges described in this Section are estimated by application the problems of end users reviewed in two previous Chapters to the case of XFormsDB.

First of all, user is required to have a sufficient knowledge of components of XFormsDB framework. Even though it consists of only declarative languages, a user who has no skills in programming on XFormsDB has to overcome a barrier of learning internal languages of XFormsDB. User is required to have knowledge of XFormsDB architecture to code even a simple application.

User has to create and keep accurate the file structure of the application and auxiliary configuration files. When the application is ready to be published online, user is required to maintain deployment environment, which consist of Web server with additional programs and libraries installed and configured. A big obstacle is that the same deployment environment is needed in order to test the application being developed, so the cost of running the application is very high.

5. Requirements

This Chapter contains information related to requirements for XIDE. As it was stated in the Section 1.2, extensive user survey is out of the scope of this Thesis. Because of that, the requirements are based on the literature review of related user surveys and user experience of using existing tools and results of persona study. Persona study aims to describe typical users of the XIDE, their level of skills and tasks to estimate possible users needs and problems to overcome. A summary of end user problems and challenges analysis is presented in Section 5.3. Finally, functional requirements designed to overcome those problems are presented together with technical requirements for the tool.

5.1. Personas

Trying to imagine system's users and their tasks and problems can help to understand better the requirements the system has. (Rosson, Ballin, & Nash, 2004) Research based on user profiles technique can help to transfer developer's ideas and thoughts about the users into official form and check if these ideas are realistic. (Kuniavsky, 2003)

According to instructions about applying the Personas technique (Kuniavsky, 2003), developers think about future system users and create *User Profiles* (or *Personas*) of the fictitious users, who will use the system. Each profile includes user's background, knowledge, tasks and problems and shows the functionality this user expects from the system.

Three user profiles were created before the design and implementation of XIDE started. Briefly they are *pure end user*, *end user* and *expert*. In order to avoid drawing attention to unnecessary details, there was no much personal information in the personas at the beginning. (Kuniavsky, 2003) (Calabria, 2004) User profiles were focused on different background skills and knowledge, different tasks to perform in the system and different way of performing those tasks. Later during the work some details were added. Since deep detailing and persona cards are needed to share personas inside a big development team, it was decided to skip those details. (Long, May 2009) Specific demographic details were added later on to make personas more descriptive and real.

Here is brief description of the user profiles:

Pure end user: Mike, Male, 30, car repairman, owner of a garage.

Usage of computer and Web: He uses a computer for surfing the Internet and paying bills. He has never developed any program or Web site. He has a strong belief that it is very difficult and he would never do it.

Tasks: He wants to create a Web page for his garage, so it can be found using search engines. He has some content to add (static info, type of work he does, prices). He needs interactive features like online-booking and responses.

End user: Alice, Female, 23, student of physics science.

Usage of computer and Web: She is familiar with computer and spends a lot of time behind it in the university and at home. She does programming for study projects (MATLAB, Java). She actively uses social media and contributes to Wikipedia. She tried to create a personal page by hand, so has general knowledge about XML, HTML and CSS. She has never tried XFormsDB.

Tasks: She wants to make a Web site about the project she did during one of the courses (e.g., static content, styling, visitor counter, news feeds, responses, questions). Also, she wants to become familiar with the XFormsDB technology.

Expert: John, Male, 30, volunteer administrator of the XIDE

Usage of computer and Web: He is familiar with computer. He has knowledge of XForms & XFormsDB; he has an experience in developing XFormsDB based Web sites.

Tasks: He is responsible for creating and updating components for XIDE. He needs to develop new components and test them in sample applications.

5.2. XIDE workflow

General idea of the XIDE tool can be presented by the description of an example workflow. This demonstrates the major functionality of XIDE and provides an overview of the system and typical activities users perform during Web application development in component-based system. This section contains a description of an imaginary workflow of Alice, one of personas described above.

Alice, a user of XIDE, invents the idea of a Web application she needs to have to share the information about her project at university. She logs into XIDE and looks through existing components and finds several ones, which suit the idea of the application. The complexity of the component can vary: it can be a simple clock, which shows current time of the corresponding time zone; it can be a complex photo gallery or a featured map. Then, Alice adds the component to the page, configures it or adds some minor changes according to her idea. She can modify existing component to add/remove some functionality or create a new component. She also can modify the source code of the page directly. Finally, Alice makes the application available in the Web so she and other people can access it.

This approach fits both pure end users and more professional end users. The whole process of creating new application can be done just by reusing components and filling forms without entering a single line of code. On the other hand, more advanced users can go deeply into the source of the component or page and edit everything manually. In addition, both pure end users and advanced end users will benefit from application management part and other general features of the system.

5.3. End users barriers

User tasks, challenges and problems discussed in background were analyzed to create a list of barriers end users meet during the development. Also, results of personas study are taken into account to fine specific problems of end users. In order to organize the resulting barriers, they are grouped based on classification from the study of analyzing barriers in programming systems described in (Ko, Myers, & Aung, 2004). Considering the scope of work described in Section 1.2, the classification is utilized in order to represent specific barriers that end users face when developing Web applications.

Table 1 End user barriers

B 1	**Design barriers:** *I don't know what I want the computer to do…*
1.1	End users are not familiar with the possibilities of the technology and do not know what can be done and how rich their application could be.
1.2	End users face difficulty in designing the Web application and detailing how to implement the functionality. Mental models of end users are different from

	professionals.
B 2	**Selection barriers:** *I think I know what I want the computer to do, but I don't know what to use…*
2.1	End users are not familiar with the technologies and tools. They do not know which technology or tool to select for their task and whether the selection allows to complete the task.
2.2	In component-based systems, users face the problem of searching for components that are appropriate for their task, check if it works and understand whether it is the one they need.
2.3	In component-based systems, users may find that there is no component that provides the functionality they need.
B 3	**Coordination barriers:** *I think I know what things to use, but I don't know how to make them work together…*
3.1	End users do not know how to combine different technologies together in order to build a Web application.
3.2	In component-based system, end users face the problem of understanding how to use the components and integrate them into the application.
B 4	**Use barriers:** *I think I know what to use, but I don't know how to use it…*
4.1	End users need to overcome an initial barrier of installation of environment in order to start the development.
4.2	End users face the problem of understanding metaphors and terms used in the tool.
4.3	End users often do not have a background in Web development, so there is a high learning barrier to overcome in order to start the development from scratch.
4.4	It is difficult to edit the source code of the application, because it is written on programming language that is not understandable and looks complicated.

4.5	It is difficult for end users to perform actual coding activities. They have a lot of syntax errors because of lack of experience and poor knowledge of language.
4.6	In component-based system, it is difficult for end users to stop using components and start editing the code, because that requires learning the background technology.
4.7	In component-based system, end users face difficulty to contribute a new component, because they are required to know the internal architecture of the system and component model.
4.8	It is difficult for end users to manage existing application sources, remember details of the application and what functionality it provides.
4.9	It is difficult to manage XFormsDB application, because it consists of several files, each of them has different type and content; they have to be combined during the development.
4.10	End users face the problem of deployment the application when it is ready, e.g., deployment the application on external server or maintain own server and database.
B 5	**Understanding and Information barriers:** *I thought I knew how to use this, but it didn't do what I expected …*
5.1	It is difficult for end users to test and debug the application, because they do not see the link between the source code of the application and the output.
5.2	It is difficult for end users to understand the external behavior of the application and manage error messages.
5.3	End users have a problem to understand internal behavior of the program, what each part does.

5.4. Requirements

The main challenge for end user is to create a working application. This task consists of overcoming smaller barriers, described in previous Section. Taking into account general idea of gentle slope of complexity, the goal of XIDE is to hide those barriers and smoothly facilitate end user during all stages of Web development process.

5.4.1. Functional requirements

This section contains most important functional requirements for XIDE that define what features the tool must provide in order to support the goal above. Requirements are organized into several groups. First group contains requirements related to management of the applications: creating, editing and publishing. Requirements in the second group define page editing functionality. Finally, third group contains requirements related to using and contribution of components.

Table 2 Application management functional requirements

FR 1	**Creation of an application and pages**
1.1	It must be possible to create an application and add pages to it using only visual interface without any manual code editing.
1.2	It must be possible to create empty application/page.
1.3	It must be possible to create an application/page based on sample application/page that include draft functionality or ready layout solutions.
1.4	It must be possible to create application/page as a copy of existing application shared by other people.
FR 2	**View and edit applications and pages**
2.1	It must be possible to view all applications and pages user has created earlier and edit any of them.
2.2	Each application/page must have descriptive information, telling when and what this application/page was created for.
FR 3	**Manage application deployment**
3.1	XIDE must provide possibility to publish the application, manage its deployment state and view whether it is published or not.
FR 4	**Example applications**

4.1	XIDE must provide example applications that can be used as a demonstration of the possibilities and usage of XFormsDB technology and XIDE.

Table 3 Page management functional requirements

FR 5	**Views of the page**
5.1	XIDE must provide a WYSIWYG-like representation of the page content.
5.2	XIDE must provide a view where the hierarchy of the page is shown.
5.3	It must be possible to view the files related to the page.
5.4	XIDE must provide a possibility to view and edit source of the page directly.
FR 6	**Support of direct editing**
6.1	XIDE must provide a support for direct editing of the source code: highlight the syntax, highlight logical structure of the editing element, check errors, and provide templates.
FR 7	**Preview and design at runtime**
7.1	XIDE must provide a preview mode that does not restrict the functionality of the application being developed.
7.2	XIDE must support design at runtime, i.e., editing the source code of the page and immediate feedback on the preview.

Table 4 Component-based architecture functional requirements

FR 8	**Database of components**
8.1	XIDE must provide a database of predefined components, which has meta information about the components (tags, rating, commenting).
8.2	There must be a search engine to search for appropriate components.
8.3	It must be possible to search for a component and try it out without damaging the Web page.
FR 9	**Components**
9.1	Components must be implemented in XFormsDB technology.
9.2	The language of the component definition must be easy to understand.
9.3	Components must be reusable, i.e., one component can be used on many pages or several times on one page.
9.4	Components must be customizable, i.e., there must be a way how a user can change appearance and behavior of components without editing its source.
FR 10	**Use of components**
10.1	XIDE must provide a visual interface for adding, managing and removing of components.
10.2	It must be possible to add components by editing the source code of the page.
10.3	It must be possible to view and edit the source code of a component.
FR 11	**Contributing new components**
11.1	XIDE must support a mechanism of contributing new components to the database. It must not require additional knowledge of architecture or models to contribute a new component.

The Table 5 shows how XIDE requirements cover end user barriers, defined in previous section.

Table 5 Relation of end user barriers and XIDE functional requirements

XIDE functional requirements

End user barriers

	1.1	1.2	1.3	1.4	2.1	2.2	3.1	4.1	5.1	5.2	5.3	5.4	6.1	7.1	7.2	8.1	8.2	8.3	9.1	9.2	9.3	9.4	10.1	10.2	10.3	11.1
1.1	X		X	X				X																		
1.2			X	X				X								X	X	X	X				X			
2.1								X								X							X			
2.2										X						X	X	X		X						
2.3												X	X			X	X	X	X	X				X	X	X
3.1																			X							
3.2									X	X								X	X	X			X	X		
4.1[13]																										
4.2									X	X																
4.3	X		X	X					X	X				X	X	X	X	X	X	X	X	X	X			
4.4									X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
4.5									X				X	X	X											
4.6													X	X	X				X	X						
4.7																X			X	X						X
4.8					X	X																				
4.9			X	X	X	X					X		X	X	X											
4.10							X																			
5.1							X		X	X	X			X	X											
5.2							X		X	X				X	X											
5.3							X		X	X				X	X	X										

5.4.2. Technical requirements

After user profiles and functional requirements have been defined, requirements related to system implementation could be defined as well. These requirements are technical issues, although they are very important for the system and its usability in general. Later on, during implementation part, these requirements are used for selection of the implementation technology.

Table 6 Technical requirements

TR1:	**XIDE must be Web-based**
	Though this requirement was defined in the scope of work, it is native to have this system Web-based. It must be possible to access XIDE from any computer. End users might not even have their own computer and access the system from public places and cannot install any applications there. XIDE requires neither

[13] This barrier influences on technical requirements

time-consuming data processing on a client nor sending a lot of information to the server so there is no strict reason why it cannot be implemented as a Web-based.

TR2: **XIDE must support direct manipulation interaction style (most important features are drag-n-drop and tree structure)**

Since the direct manipulation was decided to be the main concept of the system, it must be possible to implement drag-n-drop using selected technology. Tree structure is a native metaphor for application and containing pages.

TR3: **XIDE UI must be asynchronous**

That means that extra data must be requested from the server and loaded in the background without interfering with the display and behavior of the existing page. Using of asynchronous interface makes the system more interactive and usable. (Garrett, 2005)

TR4: **Using of XIDE by end users must not require downloading and installation additional plug-ins and libraries.**

For non-programmer, need of downloading and installing extra software can dramatically decrease system's usability and even prevent user from using the system. Moreover user can try to access to the system from public computers, where it is not allowed to install any software.

TR5: **Normal using of XIDE must be possible mostly in all modern browsers**

Since users of the system are not restricted to use any concrete browser it must be possible to use system properly at least from four most popular browsers: Internet Explorer, Mozilla Firefox, Safari, Google Chrome (Browser market share, 2010)

TR6: **It must be possible to access to XIDE mostly from all modern operating systems**

It must be possible to use the system at least from four most widespread operating systems, which are Windows XP, Windows 7, Windows Vista, Mac OS X (W3Counter, 2010)

6. Design and Implementation

In previous chapters, main features that the tool should provide are analyzed and documented as a set of requirements. The next part of this Thesis is devoted to description of constructing and evaluating a prototype tool that implements the requirements. This chapter includes description of XIDE design and implementation details.

It starts with description of tools and technologies used during the development of XIDE. After that, it presents conceptual design of the Template Language, an intermediate markup language that used to support component-based architecture. Then, important design decisions are specified. After that, first paper-based mockups of XIDE are presented in order to described initial state of XIDE user interface design.

The second part of this Chapter describes the current state of the XIDE tool. It starts with the description of final UI and features it provides. The relation of features to functional requirements is presented. Then, XIDE internal architecture is described.

6.1. Development environment

Based of technical requirements described in Section 5.4, Google Web Toolkit (GWT) was selected as a main technology for implementation of the XIDE tool. Besides it fulfills all technical issues, it allows to develop the whole Web application using one language, Java. The author of this thesis had a strong background in Java, so using of GWT eliminated learning barriers and facilitated fast prototyping.

This decision partly defined selection of other tools, frameworks and libraries used during the development. For each case it was also preferable to select popular, free, open source option with active development community.

The implementation was done partly on Windows XP and OS X operating systems. The following tools, frameworks and libraries were used locally during XIDE implementation:

Google Web Toolkit 1.4-1.7 (**GWT**) is a free, open source framework for developing high interactive client-server Web applications. Basic programming language of both client-side and server-side code is Java; on the client-side it is combined with HTML, CSS and JavaScript. Client-side Java code is automatically transformed into JavaScript during the compilation phase. The development was started with GWT version 1.4 and Java version 1.5, later it was upgraded up to GWT 1.7 and Java 1.6. In this project, GWT together with pure Java, XHTML, JavaScript, CSS, and XSLT were used to implement most of client-side and server-side functionality.

Eclipse IDE for Java EE Developers 3.4.0 with Subclipse 1.2.4 is a free, open source integrated development environment, which consist of different frameworks, features and extensions build together to support software development process. Subclipse plug-in is responsible for working with SVN revision control system. In this project, Eclipse was used as a main IDE for client- and server-side development.

Apache Subversion 1.5 is a free, open source version control system for storing and managing current and old versions of the project. During XIDE development it was used as version management system through Subclipse plug-in for Eclipse.

MySQL 5.1 is a free open source relational database management system, which runs as a server and provides a possibility to control the creation, maintenance, and the use of databases. XIDE internal database is located on MySQL server and XIDE uses MySQL queries to manage the information.

ASI 0.9[14] is a free, open source platform for social media applications developed in Aalto University, Finland. Among other functionality, it provides a common database of user accounts, which can be used in any Web application. In this project, it was used as authentication engine.

Mozilla Firefox 3.1-3.5 with **FireBug** plug-in is a free, open source Web browser, which allows to access Web sites. By means of the FireBug it is possible to inspect and edit source code of the Web page. This browser was used as a default browser for testing XIDE functionality. Initially, when this project began, XFormsDB-based applications worked correctly only in Firefox.

In addition, several external libraries were used during the development, e.g., **CodeMirror 0.64**[15], which supports dynamic highlighting of text displayed in the Web browser; **gwt-dnd**

---

[14] Aalto Social Interface http://cos.sizl.org/
[15] CodeMirror library http://codemirror.net/

2.6.5[16], a drag-n-drop library for Google Web Toolkit; MySQL connector; libraries used to work with XML.

The following environment was used to execute XForms-based applications. The same environment was installed on external server, where XIDE was published.

Apache Tomcat 5.5.27 - is a servlet container implementing the Java Servlet and Java Server Pages specifications from Sun Microsystems, Inc. Apache Tomcat is running as a base Web server for the XIDE application.

Orbeon Forms dev-post-3.7.1.200910160000 is an open source framework for developing Web forms. It is built around AJAX-based XForms engine, which allows to view XForms-based Web pages using standard browsers. In this project, it is used to transform XForms into XHTML and JavaScript, when user requests an XForms-based Web page.

eXist-db 1.2.4 is a free, open source native XML database management system. In this project, it is used by XFormsDB applications to store its data.

XFormsDB 1.0[17] is an extension to XForms framework for developing applications with advanced server-side functionality. In this project, XFormsDB engine is used to make a transformation from XFormsDB to XForms.

Additionally, the following tools were used for XIDE stress testing:

BrowserMob[18] is a Web-based solution, which provides load testing and Website monitoring services. It provides AJAX support and allows to simulate real users testing the Web site by running Silenium-based scripts.

SeleniumHQ[19] is a Web application testing system, which allows recording user actions happened in the browser as a script and then simulate user work by playing back the script. In this project, it was used to record testing scripts for stress testing.

[16] gwt-dnd library http://code.google.com/p/gwt-dnd/
[17] XFormsDB http://code.google.com/p/xformsdb/
[18] BrowserMob http://browsermob.com/performance-testing
[19] SeleniumHQ http://seleniumhq.org/

6.2. Component-based architecture

Design of XIDE consisted of two major activities. First activity was to design the concepts of the system and the mechanism how reusable abstractions are defined and utilized in the application being developed. Second activity was to design the user interface to fulfill the requirements and facilitate the usage of components and development of the application. This section describes the way how component-based approach is applied in XIDE.

From a technical point of view, a Web application consists of one or more Web pages. In case of plain XFormsDB, each page is written on XFormsDB, which is a combination of XHTML, XForms and XFormsDB tags. (Laine, 2010) In XIDE, XFormsDB page can contain *components* (see Figure 6). Each component provides some functionality, which is implemented using XFormsDB technology. So, Web page can contain mixed XFormsDB code and components. This approach supports gentle slope of complexity, because the components and the page itself are implemented on the same technology. In case if user needs to edit the source code of the component, there is no learning barrier to overcome.

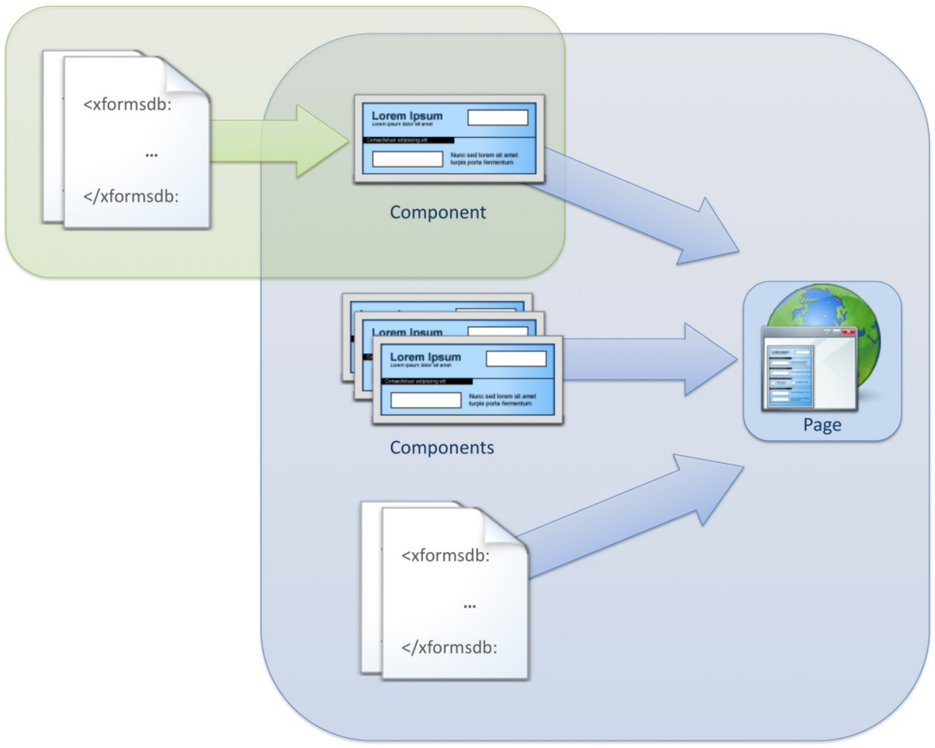


Figure 6 XIDE reusable components

According to requirements, it must be possible to adjust a component without actual coding. This requirement can be achieved by introducing parameters to components (see Figure 7). Depending on implementation and parameter type, a component can change its appearance or behavior according to values of parameters set by a user. Parameters can be also used to set up a link between to components. This allows users to configure components to some extent without any knowledge of XFormsDB technology.

XIDE components are reusable. In terms of programming language, a component is a *function*. One can define a function once in the program, and then call it from several places with different parameters. The same idea is used in XIDE components. A component is defined once in the database of XIDE, but it can be called from any Web page.

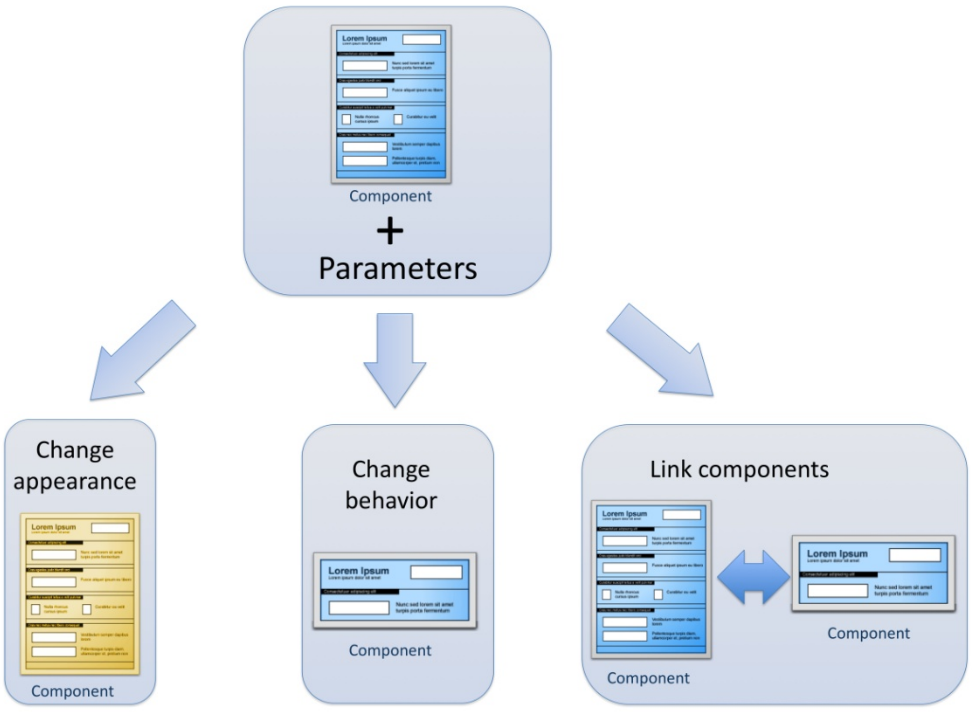


Figure 7: Customizable components in XIDE

As component is implemented on XFormsDB, it consists of the same elements as a plain XFormsDB application. In addition, component object has meta-information (component general information and description, tags, parameter definitions).

Components can be added to the page, or, according to the discussion above, called from the page source. In order to define a place, where component must be added, a special *container* element is used. Container defines a place where components can be positioned on the page.

Introducing a concept of container to the component-based architecture of XIDE brings the following benefits. A container can have graphical interface in order to show a place on the page where a component can be added. In the page source code, containers can be organized to provide different page layouts. Containers can be grouped via XFormsDB; when user will drag a components, it will be automatically placed into correct place on the page, e.g., sidebars, header or footer. A container is the only place where the component can be added, thus user cannot break the page source code by adding a component into inappropriate place. In addititon, a container can have an embedded validation mechanism, so it can check whether the component can be added inside this container.

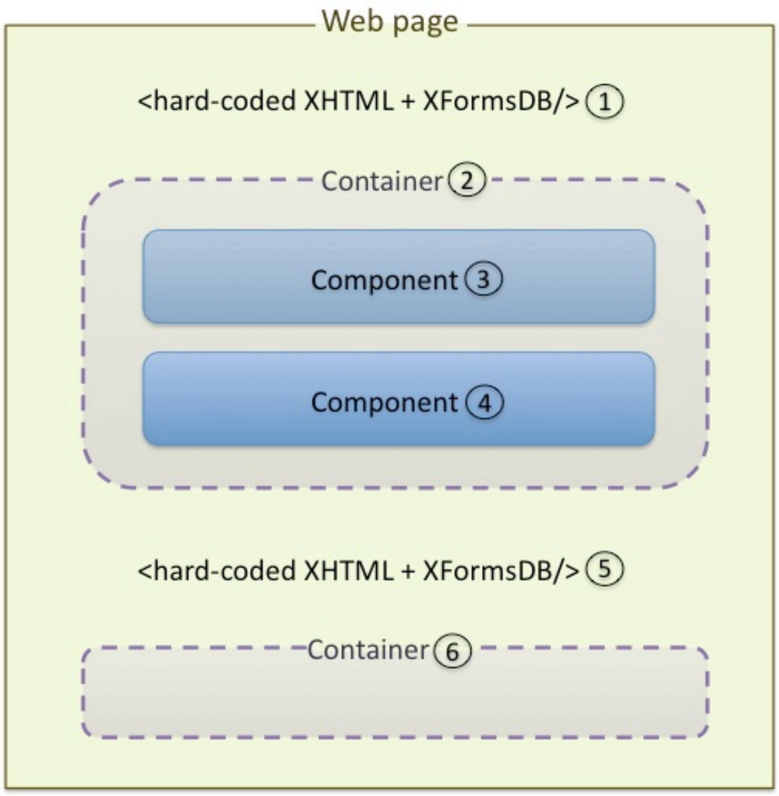

Web page
<hard-coded XHTML + XFormsDB/> ①
Container ②
Component ③
Component ④
<hard-coded XHTML + XFormsDB/> ⑤
Container ⑥

Figure 8 Concept of the Web Page

Finally, the concepts of XIDE component-based architecture are the following: Web page, contaner and component (see Figure 8). An example Web page is written on XFormsDB and two containers (② and ⑥) are built into the code. The container ② contains two component calls (③ and ④).

Component and container objects exist only during application development in XIDE. When the application published, these objects are transformed to plain XFormsDB. An intermediate declarative language, called Template Language (TL), is used to define these objects.

As it was defined in the requirements, TL must be transparent and does not require any additional skills or knowledge of technology.

TL was made as an extension of XML because of several reasons. First, TL is declarative and thus suits well for use with the XFormsDB technology. Second, it is intuitive for end users, who have some knowledge of markup language. (Cypher, Lau, Nichols, & Dontcheva, 2009) Finally, XML-based languages are successfully used for similar tasks of defining components in professional component-based Web engineering. (Yang & Papazoglou, 2006)(Gaedke & Rehse, 2000)

Except the TL, there is no internal API or library that a developer forced to use while editing existing or creating new components. That makes the modification and contribution of a new component a straightforward task, because it requires minimum additional skills or knowledge of technology. (Lieberman, Paterno, Klann, & Wulf, 2006)

6.3. User interface design decisions

The following important decisions were made in order to create a user interface that employs the requirements, facilitates end users activities and do not restrict them in controlling the system:

Decision 1: Provide several perspectives (i.e., views or screens) for different development activities. Each perspective contains information and functionality connected to the activity and does not confuse the user with unnecessary details related to other activities.

Decision 2: Consider user role for representing the functionality and information and display only information that is relevant to the user role. Practically, show high-level information initially and unhide more detailed and technical information only if user requests it.

Decision 3: Use direct manipulation for components, e.g., drag-n-drop for adding, managing, deleting components in order to lower learning barriers. (Shneiderman & Plaisant, 2010) Combine direct manipulation of visual objects with editing of the source code of the objects to facilitate transition from high-level manipulation to more powerful activities of direct editing. (Hundhausen, Farley, & Brown, 2009)

Decision 4: Provide obvious metaphors and visually represent them in the user interface, so that user can understand the relations between notions. (Shneiderman & Plaisant, 2010) (Tullis, 2005) (Barr, Biddle, & Noble, 2002)

Decision 5: Use coordinative windows. Show one concept from different points of view or using different level of abstraction depending on the purpose of the view. (North & Shneiderman) It also helps to save the screen space required to display the information about the program and reduce scrolling and searching activities. (Repenning & Ioannidou, 2006) This approach allows to reduce information access time, because a user accesses the information on the level of abstraction that is required for the task. Also, that helps to discover the link between different representations of the object, e.g., between component source code and component visual output.

Decision 6: Using well-known design concepts and ideas that are used in other popular applications, such as drag-n-drop of components, IDE-like environment, tabs, properties of the selected element, tree representation of the hierarchy, etc. That helps novice user to get familiar with the system and understand how to perform the task or get the information. (Myers, Hudson, & Pausch, 2000)

Decision 7: Provide wizards and automatic generation of source code for configuring technical issues that can be complicated for end users. (Shneiderman & Plaisant, 2010)

Decision 8: Support consistent application style and utilize design guidelines for creating a usable Web application. (Tullis, 2005) In addition to traditional consistency related things, such as having an overall standard of fonts, colors and image sizes, it is important for Web pages to be in line with well-known Web features. (Sano, 1996) Most important of these features are: use underline decoration only for links; show navigation elements all the time in the same place; provide appropriate reaction for using back and forward buttons of the browser etc. If these issues are left out, the Web page interface goes in conflict with user habits for browsing the Internet, what can lead to misunderstanding and errors. (Shneiderman & Plaisant, 2010)

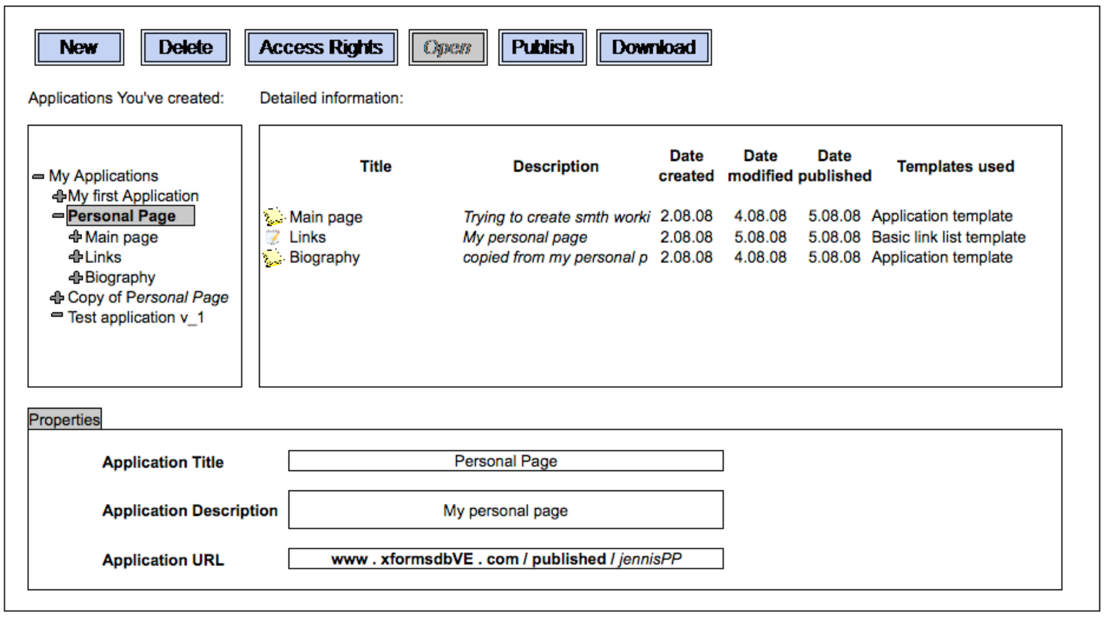
Decision 9: Provide a navigation header that shows where a user is in relation to Web site structure and what are other available options and features. In case of complex Web sites, navigation becomes a central issue. (Benyon, Turner, & Turner, 2005) A widely used standard is that Web site has top banner and navigation bar on the left. Both elements are constantly presented on each Web page, so a user always knows where to reach necessary navigation related information.

6.4. Initial design of XIDE UI

Initial design of XIDE UI was created as a set of sketches made in Microsoft Excel. These sketches were step based, showing the different system's views.

There were two views in XIDE, *Application view* and *Page view*, dedicated to two general activities of the user, managing the applications and editing the page respectively.

Application view (see Figure 9) was designed to monitor and manage the applications and pages that the user has created previously. It employed the concept of Windows Explorer in order to show the tree of existing applications and pages and their properties. It also provided an interface to edit the property of the selected application.



Applications You've created:

- My Applications
  - My first Application
  - Personal Page
    - Main page
    - Links
    - Biography
  - Copy of Personal Page
  - Test application v_1

Detailed information:

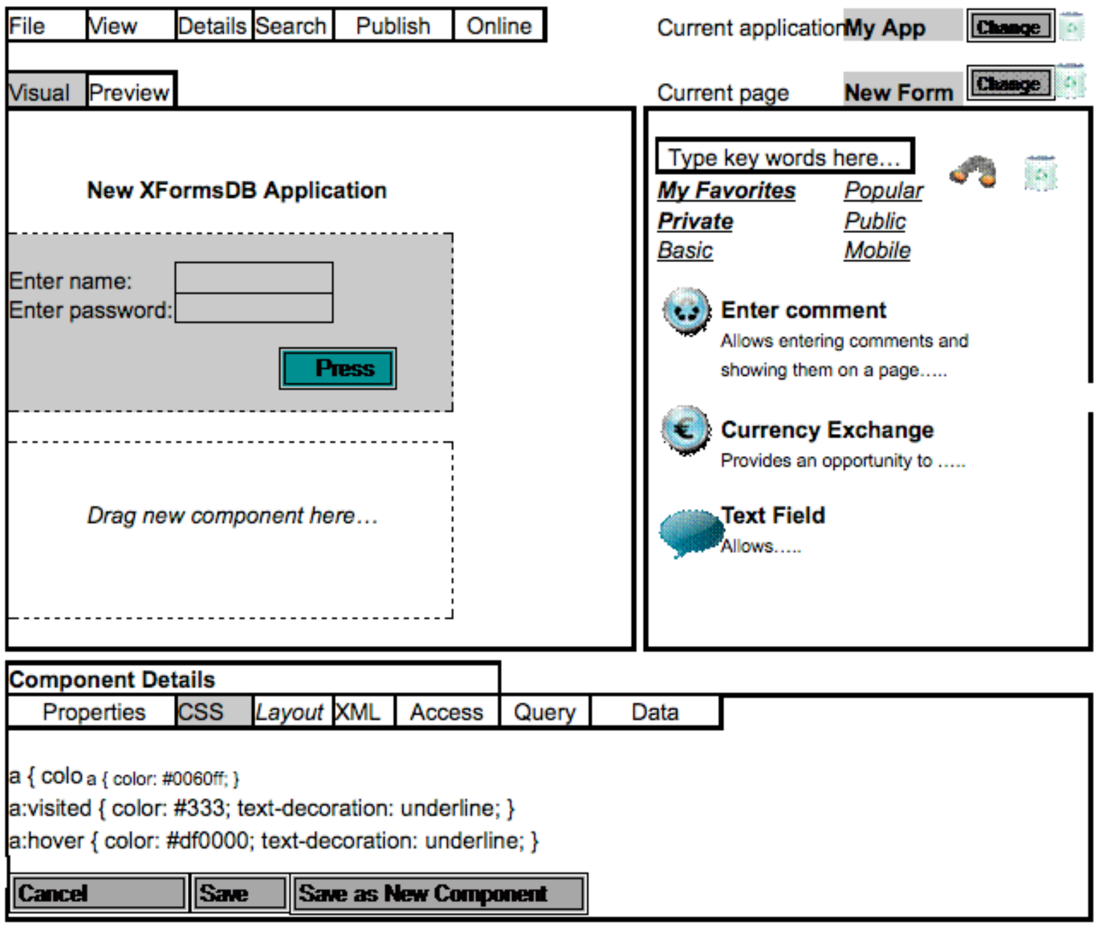
Title	Description	Date created	Date modified	Date published	Templates used
Main page	*Trying to create smth worki*	2.08.08	4.08.08	5.08.08	Application template
Links	*My personal page*	2.08.08	5.08.08	5.08.08	Basic link list template
Biography	*copied from my personal p*	2.08.08	4.08.08	5.08.08	Application template

Properties

Application Title	Personal Page
Application Description	My personal page
Application URL	**www . xformsdbVE . com / published /** *jennisPP*

Figure 9 XIDE initial design: Application view

Page view (see Figure 10) was designed to edit the content the page. It employed the conceptual design of modern IDEs, such as Eclipse IDE, where set of toolbox and views is provided to support the user during the development activities. This view provided the *Design tab*, where the content of the developing page was displayed visually. It also contained the *Properties tab*, which displayed a set of properties of the selected element, the *Search tab*, which represents list of components in the system and the *Navigation tab*, which showed the page structure as a tree.



```
a { colo a { color: #0060ff; }
a:visited { color: #333; text-decoration: underline; }
a:hover { color: #df0000; text-decoration: underline; }
```

Figure 10 XIDE initial UI design: Page view

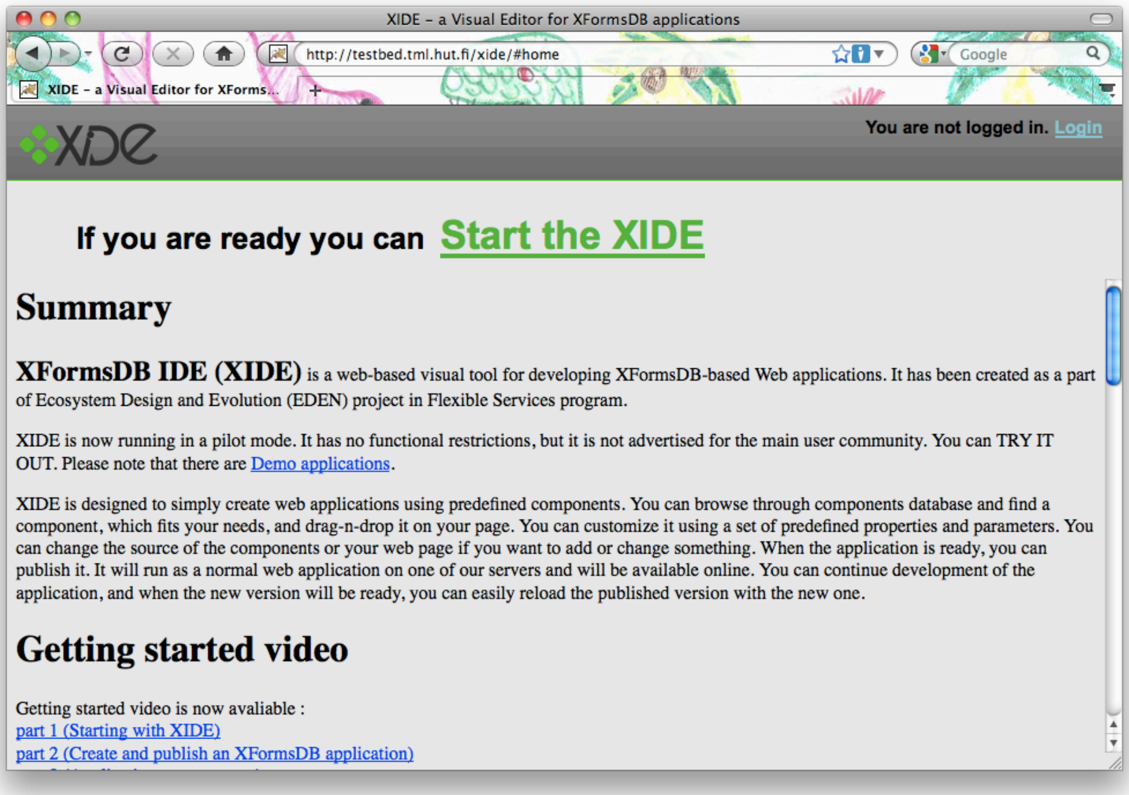
6.5. Resulting XIDE UI

The following two sections are devoted to the description of final state of the XIDE prototype tool. This section describes final UI of the XIDE tool and lists features it provides with respect to the requirements described in Section 5.4. The next section described internal architecture of XIDE implementation.

The sketches described in previous section evolved during informal process similar to Cognitive Walkthrough usability inspection method. (Kuniavsky, 2003) Although this informal testing does not bring the same results as an expert evaluation, it can give useful feedback. (Shneiderman & Plaisant, 2010) After that, the UI design and general XIDE approach were evaluated during usability testing, which is described in detail in Section 7.2.

In comparison to the initial design, final XIDE design contains several improvements. The main problem revealed by usability study was that the Explorer metaphor used in Application View was not clear for the users and thus the view was not understandable. Because there were two major tasks for this view, browsing through existing applications and managing the selected application, these two activities were separated into different views, *Application List view* and *Application view*.

Additionally, other improvements were designed and implemented. For example, the source code editing support was improved to fit user expectations. Unified mechanism of monitoring unsaved changes was added to improve reliability of the XIDE tool. Unified style of wizards and forms was developed based on results of the testing.


XIDE – a Visual Editor for XFormsDB applications
http://testbed.tml.hut.fi/xide/#home
Google
XIDE – a Visual Editor for XForms...
XIDE
You are not logged in. Login

If you are ready you can Start the XIDE

Summary

XFormsDB IDE (XIDE) is a web-based visual tool for developing XFormsDB-based Web applications. It has been created as a part of Ecosystem Design and Evolution (EDEN) project in Flexible Services program.

XIDE is now running in a pilot mode. It has no functional restrictions, but it is not advertised for the main user community. You can TRY IT OUT. Please note that there are Demo applications.

XIDE is designed to simply create web applications using predefined components. You can browse through components database and find a component, which fits your needs, and drag-n-drop it on your page. You can customize it using a set of predefined properties and parameters. You can change the source of the components or your web page if you want to add or change something. When the application is ready, you can publish it. It will run as a normal web application on one of our servers and will be available online. You can continue development of the application, and when the new version will be ready, you can easily reload the published version with the new one.

Getting started video

Getting started video is now avaliable :
part 1 (Starting with XIDE)
part 2 (Create and publish an XFormsDB application)

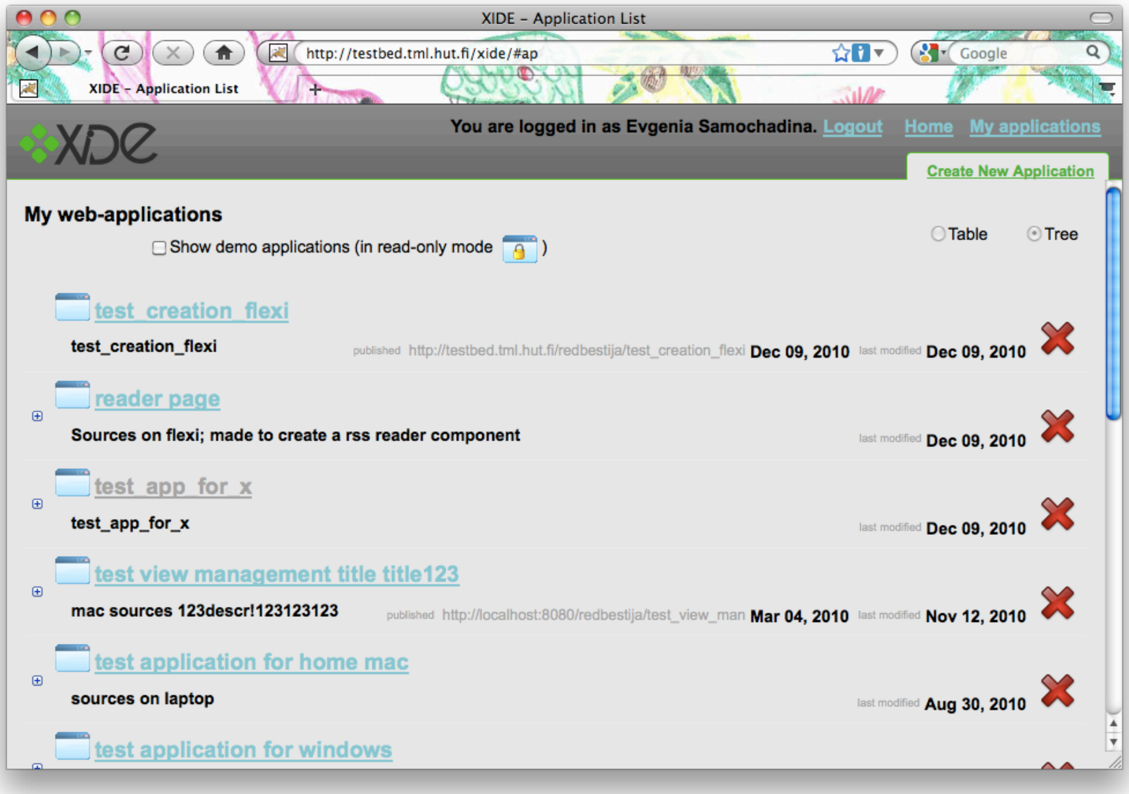
Figure 11 XIDE final UI: Welcome Page view

Finally, the XIDE tool has four views. The first view, *Welcome page view,* contains general information about XIDE, description of demo applications, introduction videos and links to other help resources (see Figure 11).

When user logs into XIDE, the *Application List view* is loaded (see Figure 12). It displays the list of applications user has created previously. This view provides the following features:

Table 7 Application List view features

F 1	**Application List view features**
1.1	View all applications and pages created by the user and related information, e.g., titles, descriptions, when was created, whether is published or not.
1.2	Access to *demo applications*, which are read-only ready-made applications that demonstrate the functionality of XFormsDB language and XIDE.
1.3	Create a new application using a form-based wizard that requires no programming activities from the user.
1.4	Create an application as a copy of existing application shared by other people.



You are logged in as Evgenia Samochadina.

My web-applications

Show demo applications (in read-only mode)

test_creation_flexi
test_creation_flexi — published http://testbed.tml.hut.fi/redbestija/test_creation_flexi Dec 09, 2010 — last modified Dec 09, 2010

reader page
Sources on flexi; made to create a rss reader component — last modified Dec 09, 2010

test_app_for_x
test_app_for_x — last modified Dec 09, 2010

test view management title title123
mac sources 123descr!123123123 — published http://localhost:8080/redbestija/test_view_man Mar 04, 2010 — last modified Nov 12, 2010

test application for home mac
sources on laptop — last modified Aug 30, 2010

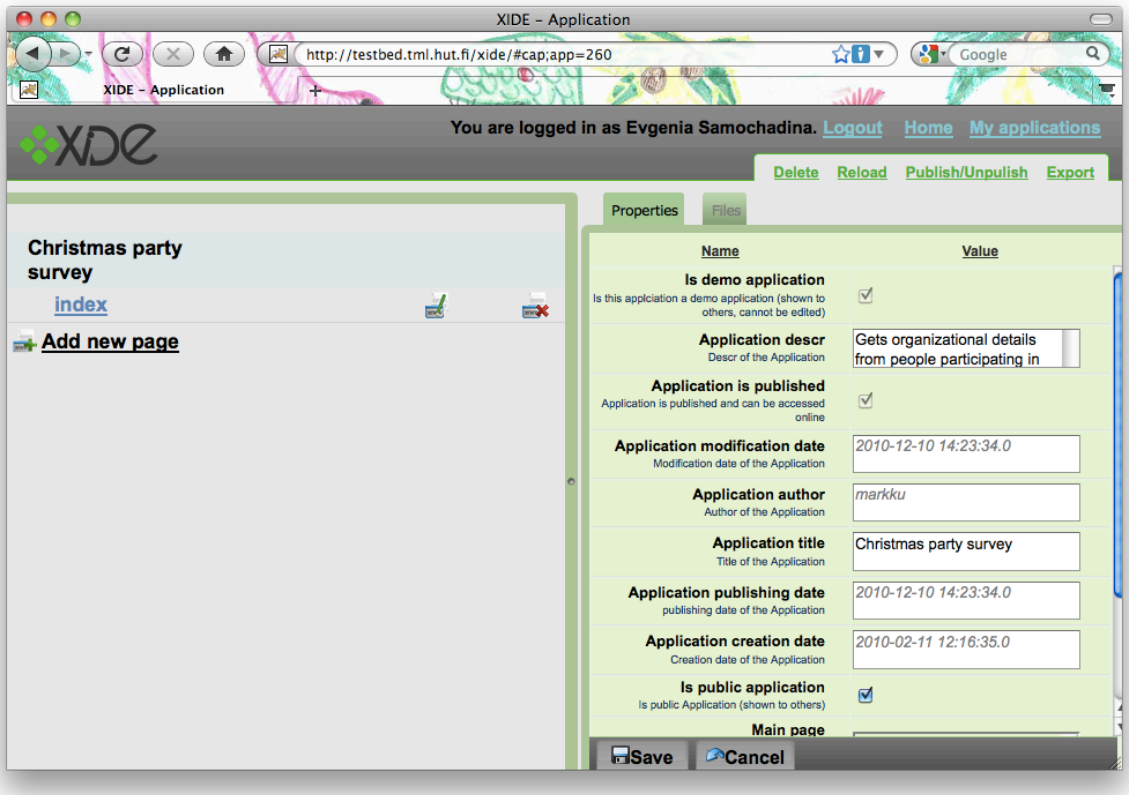
test application for windows

Figure 12 XIDE final UI: Application List view

When user selects the application to edit, the *Application view* is loaded (see Figure 13). It displays the structure of the selected application, allows to edit properties of application and containing pages and provides means for publishing management of the application. This view provides the following features:

Table 8 Application view features

F 2	**Application view features**
2.1	Edit application and page information, such as title, description, a welcome page of the application
2.2	Create a new page using a form-based wizard that requires no programming activities from the user. User may choose to create an empty page, a page with basic XFormsDB skeleton or a page with container for adding new components right away.
2.3	Perform application publishing management, i.e., publish the application, unpublish the application, and reload the application with the new version. The application is published on the internal XIDE server without any technical actions from the user.


XIDE – Application

http://testbed.tml.hut.fi/xide/#cap;app=260

You are logged in as Evgenia Samochadina. Logout Home My applications

Delete Reload Publish/Unpublish Export

Christmas party survey
index
Add new page

Properties Files

Name	Value
Is demo application Is this applciation a demo application (shown to others, cannot be edited)	✓
Application descr Descr of the Application	Gets organizational details from people participating in
Application is published Application is published and can be accessed online	✓
Application modification date Modification date of the Application	2010-12-10 14:23:34.0
Application author Author of the Application	markku
Application title Title of the Application	Christmas party survey
Application publishing date publishing date of the Application	2010-12-10 14:23:34.0
Application creation date Creation date of the Application	2010-02-11 12:16:35.0
Is public application Is public Application (shown to others)	✓
Main page	

Figure 13 XIDE final UI: Application view

When user selects the page to edit, the *Page view* is loaded (see Figure 14). It displays the structure of the page and supports page edition by different means. This view provides the following features:

Table 9 Page view features

F 3	**Page view features**
3.1	View and edit the page content visually in a schematic manner on the *Design tab*, so all UI elements of the page and components are displayed.
3.2	View the page in a full-functional preview mode on the *Preview tab*, which shows final version of the page with all functionality.
3.3	Modify the source code of the page and see the results on the Design tab and Preview tab instantly.
3.4	Edit the source code of the page and component in advanced editor that

supports highlighting of XFormsDB syntax and structure of the element.

3.5 View and manage the hierarchy of the page elements (containers and components) in a tree structure on the *Navigation tab*.

3.6 View and manage the hierarchy of component and page files and folders on the *File tab*. Edit the content of different source files on the corresponding tabs (e.g., *CSS tab*, *Queries tab*, *Data Instances tab*).

3.7 Search for a component in the database using different search criteria, such as text string, tags.

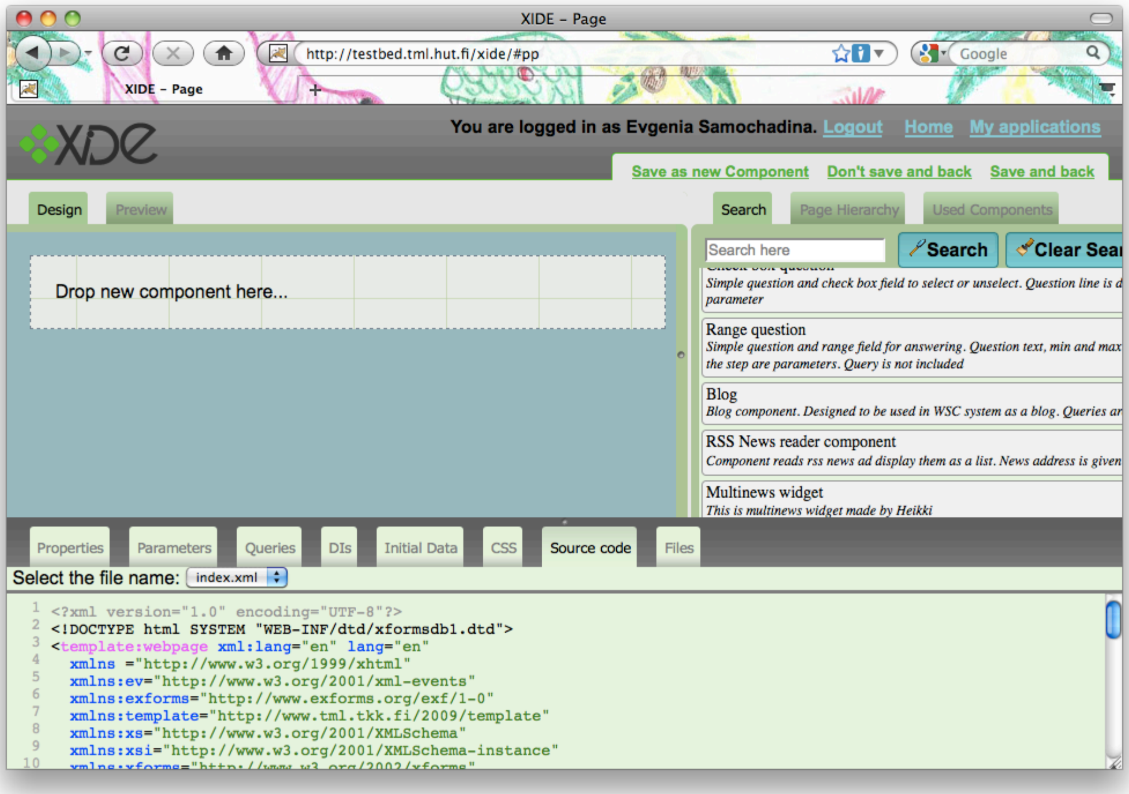
3.8 Add, manage and remove components to the page by drag-n-drop or by manual coding.

3.9 Configure component parameters in visual interface, using wizard or by manual coding.

3.10 Modify the component source and contribute a new component to the database.

3.11 Save the Web page as a new complex component to the database.

3.12 All changes that are made in the view can be saved or reverted to previous state.



You are logged in as Evgenia Samochadina.

Simple question and check box field to select or unselect. Question line is d
parameter

Range question
Simple question and range field for answering. Question text, min and max
the step are parameters. Query is not included

Blog
Blog component. Designed to be used in WSC system as a blog. Queries ar

RSS News reader component
Component reads rss news ad display them as a list. News address is given

Multinews widget
This is multinews widget made by Heikki

Select the file name: index.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html SYSTEM "WEB-INF/dtd/xformsdb1.dtd">
<template:webpage xml:lang="en" lang="en"
  xmlns ="http://www.w3.org/1999/xhtml"
  xmlns:ev="http://www.w3.org/2001/xml-events"
  xmlns:exforms="http://www.exforms.org/exf/1-0"
  xmlns:template="http://www.tml.tkk.fi/2009/template"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xforms="http://www.w3.org/2002/xforms"
```

Figure 14 XIDE final UI: Page view

The Table 10 shows how XIDE functionality covers the requirements defined in previous chapter.

Table 10 Relation of XIDE functional requirements and XIDE features

XIDE features

XIDE functional requirements

	1.1	1.2	1.3	1.4	2.1	2.2	2.3	3.1	3.2	3.3	3.4	3.5	3.6	3.7	3.8	3.9	3.10	3.11	3.12
1.1			X			X													
1.2			X			X													
1.3			X	X		X													
1.4			X	X		X													
2.1	X				X														
2.2	X				X														
3.1							X												
4.1		X																	
5.1								X											
5.2												X							
5.3													X						
5.4										X	X								
6.1										X	X						X		
7.1									X										
7.2								X	X	X									
8.1															X				
8.2														X	X				
8.3														X	X	X			
9.1																	X		
9.2																	X		
9.3														X	X	X			
9.4															X	X			
10.1												X			X				
10.2															X				
10.3									X				X				X		
11.1																		X	

6.6. XIDE architecture

Documenting software architecture is useful during all steps of project development.(Bass, Clements, & Kazman, 2003) Descriptive documenting is important to further comprehension and maintenance of the system. Also, architecture documentation helps to introduce the system to new users. (Clements, Garlan, Little, Nord, & Stafford, 2003) System architecture should be described with appropriate level of details and well organized, so the reader can easily understand the system on the required level. (Bass, Clements, & Kazman, 2003)

IEEE Recommended practices for architectural description claims, that it is highly important to identify target stakeholders, their roles and the purpose of the documentation. (IEEE Standard 1471-2000, 2000) In this Thesis, the aim of documentation is to present XIDE architecture to provide a general understanding of how the system works. There is no need to include detailed system architecture here, since it is not supposed to use this Thesis as a reference document for further development or maintenance of XIDE.

More detailed description of XIDE architecture and implementation can be found at XIDE project page[20].

Since the standard does not define exact language for architecture description, there are many frameworks used for documenting the architecture. (Bass, Clements, & Kazman, 2003) They specify the set of views that should be used. View is a representation of the system from some perspective, e.g., how the source code is organized or how the system is distributed among computers in the network. However, the frameworks assume too detailed level of documentation. Finally, it was decided to utilize general practices, described in (Clements, Garlan, Little, Nord, & Stafford, 2003). According to that, there is no fixed set of views, which should be used for system architecture description. (Clements, Garlan, Little, Nord, & Stafford, 2003) provide a classification of views and recommendations, how to choose appropriate set of them.

There are three general points of view (viewtypes) on the system architecture: Module viewtype shows the structure of the system; Component-to-connector describes how does system behaves in runtime; Allocation viewtype shows the relation between software elements and external environment. Each viewtype can contain several views. IEEE standard recommends that the view should correspond only to one viewtype in order to reduce details and simplify the perception of the view. (IEEE Standard 1471-2000, 2000)

In this Thesis, each of the three viewtypes will be represented with one view.

[20] XIDE http://code.google.com/p/xformsdb-ide/

6.6.1. Uses view

Module viewtype is represented by Uses module view. It shows logical structure of the system and dependencies between modules. Elements of the view are modules, which are units of implementation.

In this Thesis, modules are created based on implementation packages. For reader's convenience, the module diagram is separated into two parts: client- and server-side.

First, client-side uses view is depicted on Figure 15. Client part of XIDE is responsible for UI representation and managing user actions. It sends requests to the server in response to user activity and processes server response.

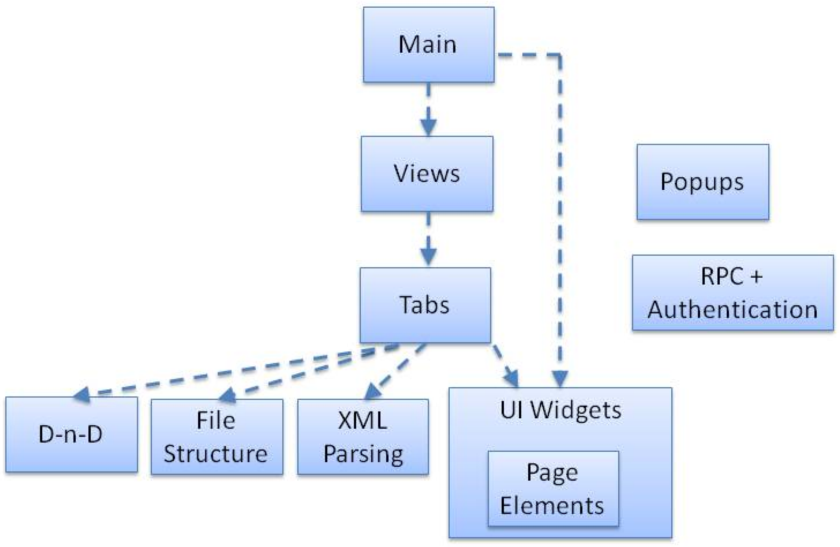


Figure 15 Uses view of the client-side of XIDE

Table 11 Uses view of the client-side of XIDE: modules description

Main	Main class takes care about XIDE client-side initialization process. After that, it manages different application views, switching to corresponding view according to both XIDE user actions and browser back and forward events. It is also responsible for corresponding header links management. Finally, it handles events, which are sent by different elements and needs

	to be propagated.
Views	Contains set of Views classes. Each of them is responsible for one view (Welcome page, Application List, Application, Page). Common functionality is event propagation management. Each view initiates the tabs it consists of and displays them on the screen accordingly.
Tabs	Tabs package contains set of different tabs. Their common functionality is possibility to update the displayed information according to received event. Each tab is responsible for rendering the information it contains and update it according to the events received.
UI Widgets	UI widgets package contains different custom UI objects to be used on the Tabs (e.g., Panels, styled buttons, etc.).
Page Elements	Page elements are Component, Container and Web Page. Each element is UI representation of corresponding Template Language abstraction and takes care about both logic and graphical representation of the object.
D-n-d	D-n-D package contains classes responsible for drag-n-drop process. It is utilized by several tabs, which have drag-n-drop functionality.
File Structure	File structure's main responsibility is to provide abstractions, which represent physical files and folders in XIDE. It is utilized by File Structure Tab, which shows physical file structure of the object at the server (e.g., application or component).
XML parsing	XML parsing takes care about parsing and rendering XFormsDB on the client-side.
Popups	Popups package contains custom popup windows with different styles and purposes. There are general popups (e.g., Error Popup or Notification popup), which are widely used in the system. There are also custom forms and wizards; they are used in special cases, e.g., New Application Wizard

	or Tag Search popup.
RPC + Authentication	RPC is responsible for communication with the server by means of Remote Procedure Call; authentication process is integrated with RPC.

Server-side implementation consists of three servlets, which are responsible for the communication with the client, and several supporting modules (see Figure 16). Main server part responsibilities are communication with the database, managing file structure, publishing the application, etc.

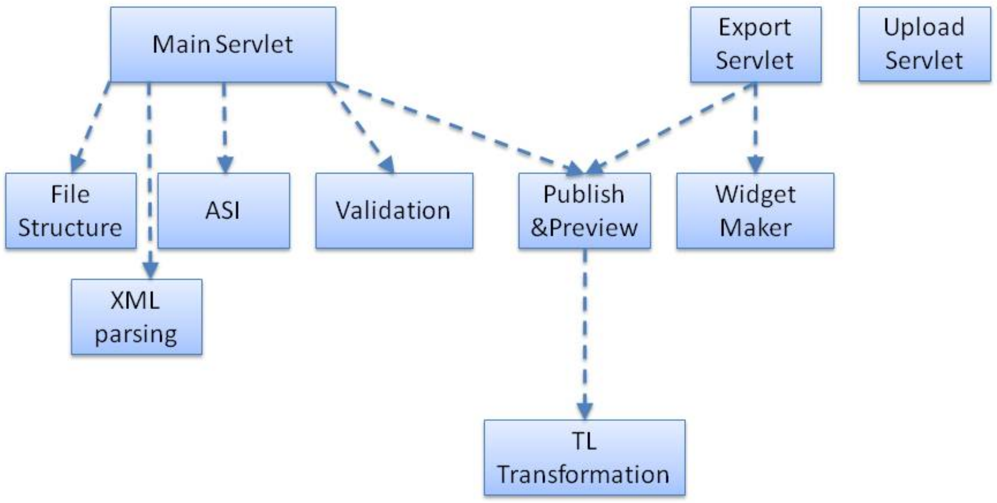


Figure 16 Uses view of the server-side of XIDE

Table 12 Uses view of the server-side of XIDE: modules description

Main Servlet	Main Servlet is responsible for communication with client based on RPC calls. It process requests to the database and forwards requests to other modules.
File Structure	This module is responsible for creating and parsing file structure on the server. It is used when new application is created or when page is requested from the client.

ASI	The module takes care about communication with ASI server to perform authentication-related tasks.
Validation	This module validates pages and components syntax. It is used when page or component is saved.
Publish & Preview	Publish & Preview module is responsible for application publishing and preview. It manages transformation, copies necessary files, initializes Exist database and deploys the application.
TL transformation	This module takes care about transformation from internal Template Language into XFormsDB. This includes substitution of parameters and component's calls with valuable source code.
Export Servlet	This Servlet handles client requests to download files from server. It is used to export the application as a widget.
Widget Maker	This module is responsible for creating archive with the published application. This archive can be used to deploy the application on the external server.
Upload Servlet	This Servlet processes client requests to save a new file to the server.

6.6.2. Client-server view

Component-to-container viewtype is represented by client-server view (see Figure 17). It illustrates how client and server communicate in the runtime. Elements in this view are client and server components and protocol connectors.

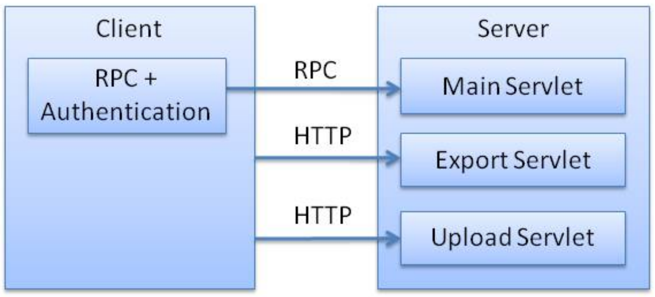


Figure 17 Client-server view of XIDE

Table 13 Client-server view of XIDE: elements description

Client	Client part of application running on user's Web browser on user's computer.
Server	Server part of the application running on the external server. Three servlets are responsible for communication with the client.
RPC + Authentication	This module is responsible for communication with the server by means of RPC. All other modules use this module in order to send an RPC request to server and receive a response.
Main Servlet	This Servlet manages all RPC requests, received from the client.
Export Servlet	This Servlet processes client HTTP requests to download a file.
Upload Servlet	This Servlet processes client HTTP POST request to save a file.
RPC	RPC is a communication technology used for remote invocation. In XIDE it is used to implement asynchronous AJAX request to the server to perform some server-side action. Often it includes communication with the database and/or managing file system.

HTTP	A standard HTTP request.

6.6.3. Deployment view

Allocation viewtype is represented by deployment view (see Figure 18). It shows how the system is distributed among the network during deployment. According to (Clements, Garlan, Little, Nord, & Stafford, 2003), it should be used for performance evaluation and improving. In this Thesis, it is more high-level and only describes how the XIDE tool is deployed and its environment.

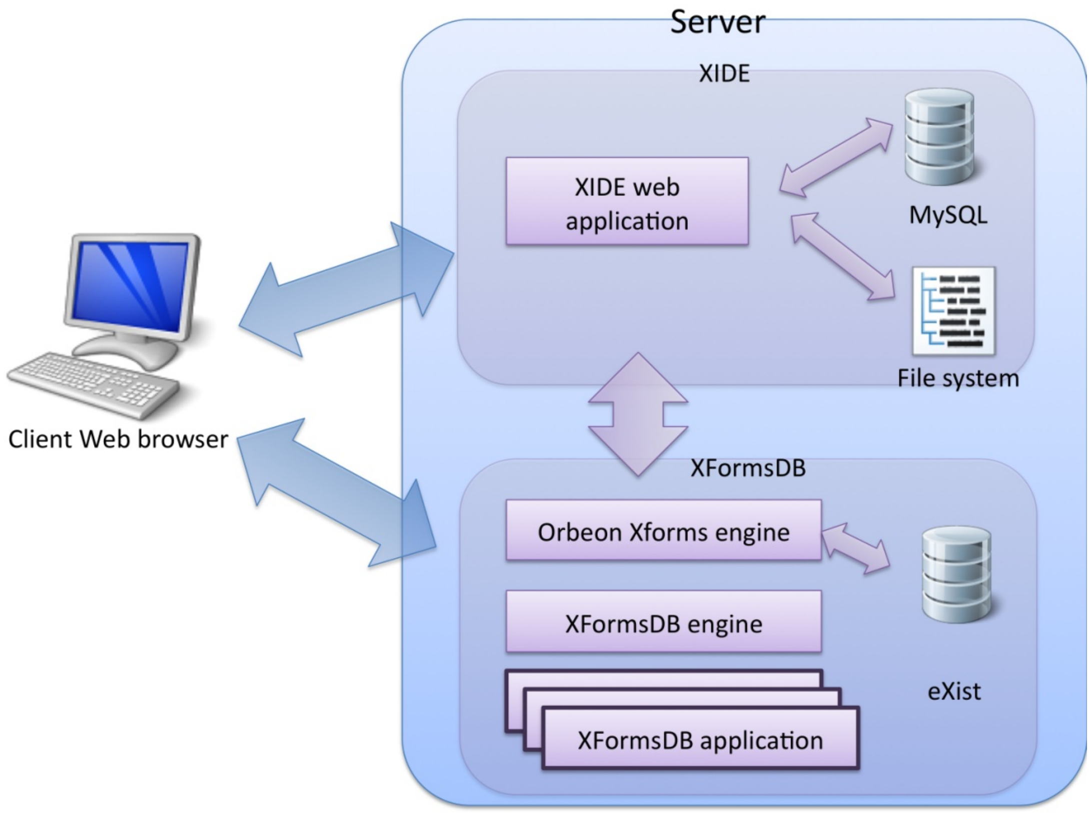


Figure 18 Deployment view of XIDE

Table 14 Deployment view of XIDE: elements description

Client Web browser	User accesses XIDE from a browser on user's computer. Also, user can access a published XFormsDB application directly.
Server	Apache Tomcat Web server supports both XIDE and XFormsDB engine.

	However, it is not obligatory to have both applications running on the same server.
XIDE	XIDE is deployed as a Web Application archive (WAR). It contains Web pages, related pictures, JavaScript scripts, and servlets to process server-side functionality. It communicates with XIDE database (MySQL) and XIDE files (File system). While XIDE processes user action, it may require communicating with XFormsDB engine. This happens when user requests to view the page, written in Template Language, i.e., when user requests to preview the page.
MySQL	MySQL relational database contains information about components, users, Web pages and applications. It also supports tags; currently tag can be assigned only to a component.
Files system	XIDE physical files include components' and application sources and published and previewed applications. They are organized and stored on the server.
XFormsDB	This part represents XFormsDB related part of the system.
XFormsDB engine	XFormsDB engine is a Web application deployed on the server. It takes care about processing XFormsDB applications. (Laine, 2010) Its main responsibilities are processing of XFormsDB into XForms transformation, communication with Exist database and processing requests to display XForms applications.
XFormsDB applications	XFormsDB applications are deployed to the server and can be accessed both by XIDE and user's browser. In order to be displayed in the browser, each application should be processed by XFormsDB engine. (Laine, 2010)
eXist	eXist XML database is used by XFormsDB applications to store the data. XQuery and XPath can be used to manipulate the data.

7. Evaluation and discussion

This section contains evaluation of the XIDE prototype tool. The evaluation starts with the presentation of the sample application and discussion whether XIDE facilitates end user development and provides a gentle slope of complexity. After that, a description of usability testing study and its results are presented and discussed. Finally, the results of expert evaluation of XIDE are presented.

7.1. Sample Web application

In this section, XIDE is evaluated by the mean of detailed inspection of the creation process of sample application. The goal is to analyze whether XIDE provides a gentle slope of complexity and facilitates end users during the development. The inspection focuses on the most important steps of the development of sample Web application in XIDE. Finally, the estimated slope of complexity for XIDE is discussed.

This section describes the process of creating a Web application by Alice, one of XIDE personas, defined in Section 5.1. A use case for the demo application is the following. Alice is a volunteer at university radio station and her responsibility is to create a list of the latest news headers and to read them during a short morning broadcast. Previously she used to look through different news Web sites stored as bookmarks in her browser and write down all interesting news headers on the paper. Now, she is going to create a Web application, which allows her to combine together all required news Web sites and an electronic notebook, where she can store the headers. She will be able to see all news feeds on one page and store an interesting header in a second; when finished browsing, she can read the resulting list of headers from the same application.

During the application development using the XIDE tool, Alice performs several actions described below.

Understanding the possibilities of the XIDE tool

Besides help information, XIDE provides demo applications, that can be viewed in order understand the possibilities of the tool and imagine what kind of applications it is possible to create with the technology. Any XIDE user can access demo applications from the

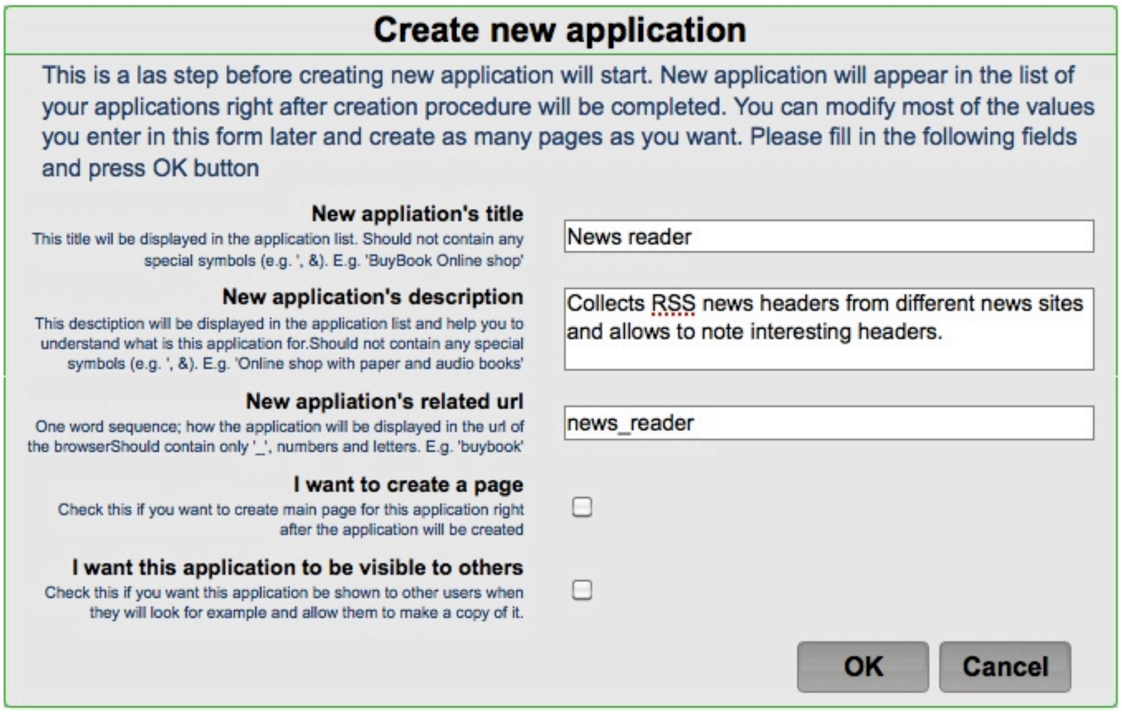
Application List view, described in Section 6.5. A user can view, modify and preview the changes, however it is not possible to save the changes.

Getting started

XIDE does not require any installation or additional configuration before a user can start using it. It also does not require installing any plugins to the Web browser. It is available online and Alice can used from any computer.

Create the application

XIDE provides a form-based wizard that allows to create a new application without any manual actions, such as creating folder structure, creating configuration files and code writing (see Figure 19). Alice does not need any knowledge about the structure of the XFormsDB application.



Create new application

This is a las step before creating new application will start. New application will appear in the list of your applications right after creation procedure will be completed. You can modify most of the values you enter in this form later and create as many pages as you want. Please fill in the following fields and press OK button

New appliation's title This title wil be displayed in the application list. Should not contain any special symbols (e.g. ', &). E.g. 'BuyBook Online shop'	News reader
New application's description This desctiption will be displayed in the application list and help you to understand what is this application for.Should not contain any special symbols (e.g. ', &). E.g. 'Online shop with paper and audio books'	Collects RSS news headers from different news sites and allows to note interesting headers.
New appliation's related url One word sequence; how the application will be displayed in the url of the browserShould contain only '_', numbers and letters. E.g. 'buybook'	news_reader
I want to create a page Check this if you want to create main page for this application right after the application will be created	☐
I want this application to be visible to others Check this if you want this application be shown to other users when they will look for example and allow them to make a copy of it.	☐

Figure 19 XIDE: Create new application wizard

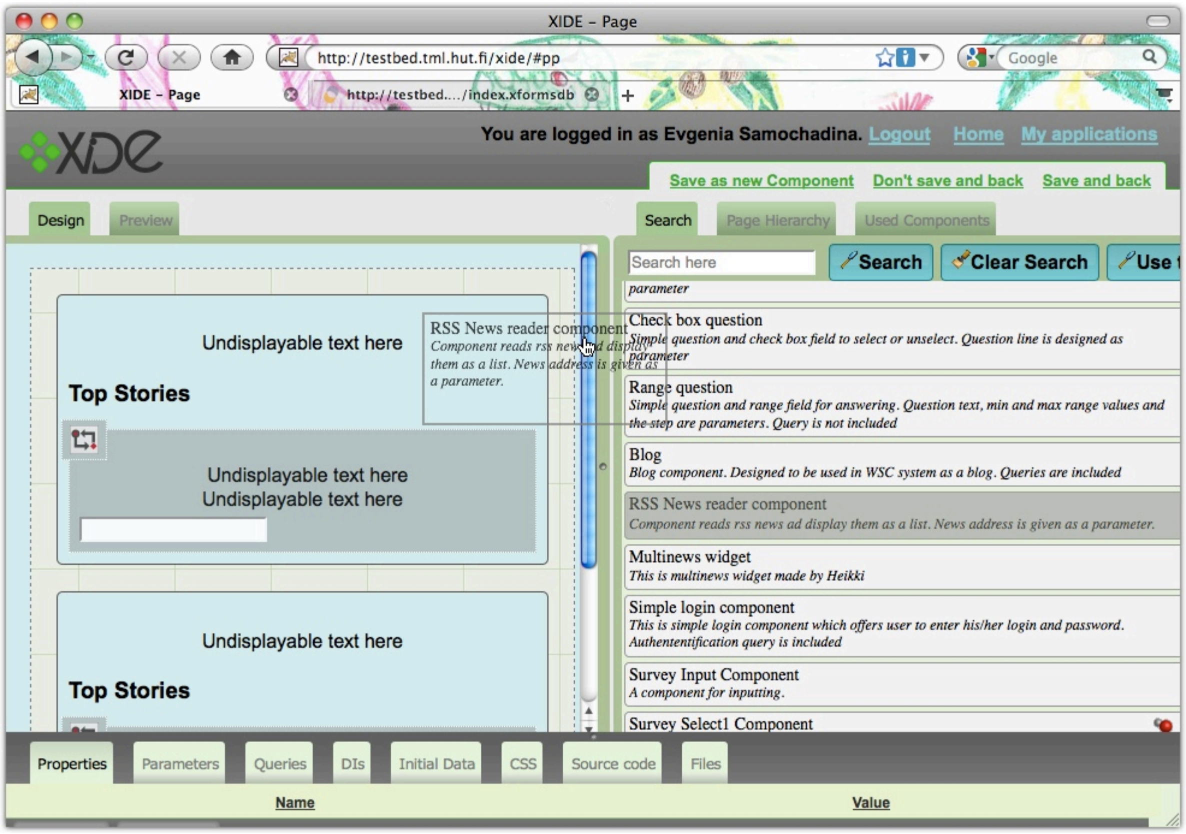
Create the page

XIDE also provides a similar form-based wizard for creating a new page. The wizard provides different options, including creating an empty page, creating a page with XFormsDB skeleton, and creating a page with a container for adding components right away. Alice decides to select the third option, because she wants to try XIDE and does not want to do any manual coding yet.

Alice creates a page and opens it in the *Page view,* described in Section 6.5.

Choose components

Alice looks for appropriate components in the component database. She tries to search for “news”, “feeds” and “notes”. Alice checks the descriptions and sees that some components are marked with “self-sufficient” tag. That means that they do not need any other components or configurations to work. She decides to use this tag in search as well. She drags some components that fit her search criteria to the page to see how they look like (see Figure 20).



You are logged in as Evgenia Samochadina.

RSS News reader component
Component reads rss news ad display them as a list. News address is given as a parameter.

parameter

Check box question
Simple question and check box field to select or unselect. Question line is designed as parameter

Range question
Simple question and range field for answering. Question text, min and max range values and the step are parameters. Query is not included

Blog
Blog component. Designed to be used in WSC system as a blog. Queries are included

RSS News reader component
Component reads rss news ad display them as a list. News address is given as a parameter.

Multinews widget
This is multinews widget made by Heikki

Simple login component
This is simple login component which offers user to enter his/her login and password. Authententification query is included

Survey Input Component
A component for inputting.

Survey Select1 Component

Figure 20 XIDE: Adding new component to the page

Finally, she decides to use two components: RSS reader and Notes components. She drags two RSS readers and one Notes component to the page. The page she created does not have any layout, so the news feeds and the note components are displayed vertically one after another.

RSS reader

Displays RSS feed as a list of clickable headers. Feed link to display is a parameter that can be edited by user.

Notes

Allows you to enter your notes and see and delete previously entered notes. Number of notes and notebook title are parameters.

Customizing components

Alice configures the RSS reader components by setting the RSS feed URLs of Web sites, which she previously used for gathering the news.

She tries the application in *Preview tab*, checks that it works as she planned and proceeds to publishing. Alice publishes the application by pressing a single button and finally gets the working application online.

Edit the page source code

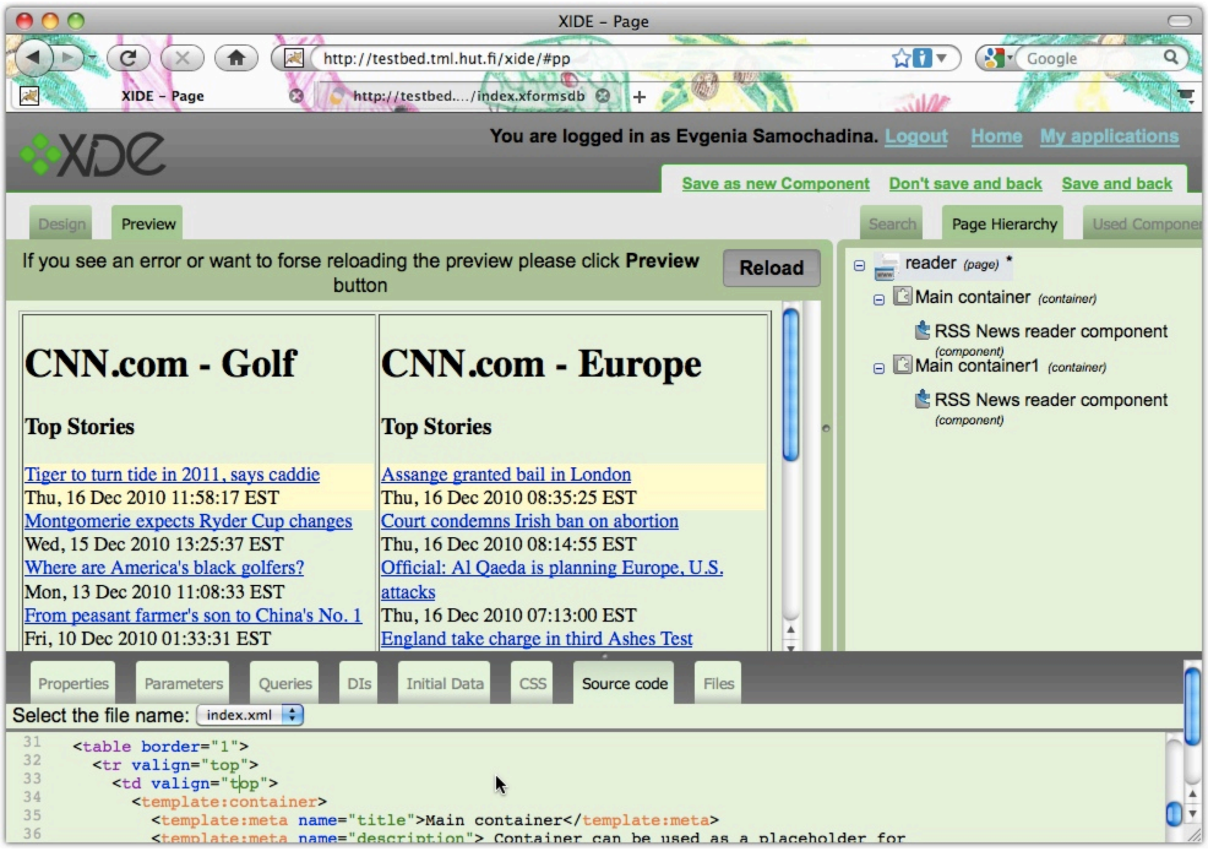
Alice uses the application for several days and finally decided to modify it. She does not like that the components are placed vertically, because it is inconvenient to scroll the page to reach her notes.

She logs into XIDE and looks for the application she created previously. She decides to create a table layout for the developing page, so that several components could be places in a row.

In order to achieve this task, she needs to make changes to the source code of the page. She selects the page visually and opens the tab with the source code. The tab provides advanced code editing features, such as highlighting of XFormsDB and TL syntax and error checking on the fly.

Preview the page and page hierarchy

In order to see the final result of her coding, Alice decides to open the *Preview tab* while she does the code editing (see Figure 21). When she changes the source code, the changes immediately appear on the preview of the page. She also checks the structure of the page on the *Page Hierarchy tab*.


XIDE – Page

http://testbed.tml.hut.fi/xide/#pp

XIDE – Page

http://testbed..../index.xformsdb

You are logged in as Evgenia Samochadina. Logout Home My applications

Save as new Component Don't save and back Save and back

Design Preview

Search Page Hierarchy Used Compone

If you see an error or want to forse reloading the preview please click **Preview** button

Reload

reader (page) *
Main container (container)
RSS News reader component (component)
Main container1 (container)
RSS News reader component (component)

CNN.com - Golf

Top Stories

Tiger to turn tide in 2011, says caddie
Thu, 16 Dec 2010 11:58:17 EST
Montgomerie expects Ryder Cup changes
Wed, 15 Dec 2010 13:25:37 EST
Where are America's black golfers?
Mon, 13 Dec 2010 11:08:33 EST
From peasant farmer's son to China's No. 1
Fri, 10 Dec 2010 01:33:31 EST

CNN.com - Europe

Top Stories

Assange granted bail in London
Thu, 16 Dec 2010 08:35:25 EST
Court condemns Irish ban on abortion
Thu, 16 Dec 2010 08:14:55 EST
Official: Al Qaeda is planning Europe, U.S. attacks
Thu, 16 Dec 2010 07:13:00 EST
England take charge in third Ashes Test

Properties Parameters Queries DIs Initial Data CSS Source code Files

Select the file name: index.xml

```
<table border="1">
  <tr valign="top">
    <td valign="top">
      <template:container>
        <template:meta name="title">Main container</template:meta>
        <template:meta name="description"> Container can be used as a placeholder for
```

Figure 21 XIDE: Preview of the page and hierarchy of the page

Finally Alice gets the improved application published.

Edit the component

Alice showed her application to her group mate, who argued that the style of components is not good. Alice decides to try to change the style. She thinks she became more familiar with the XFormsDB while she was creating the application.

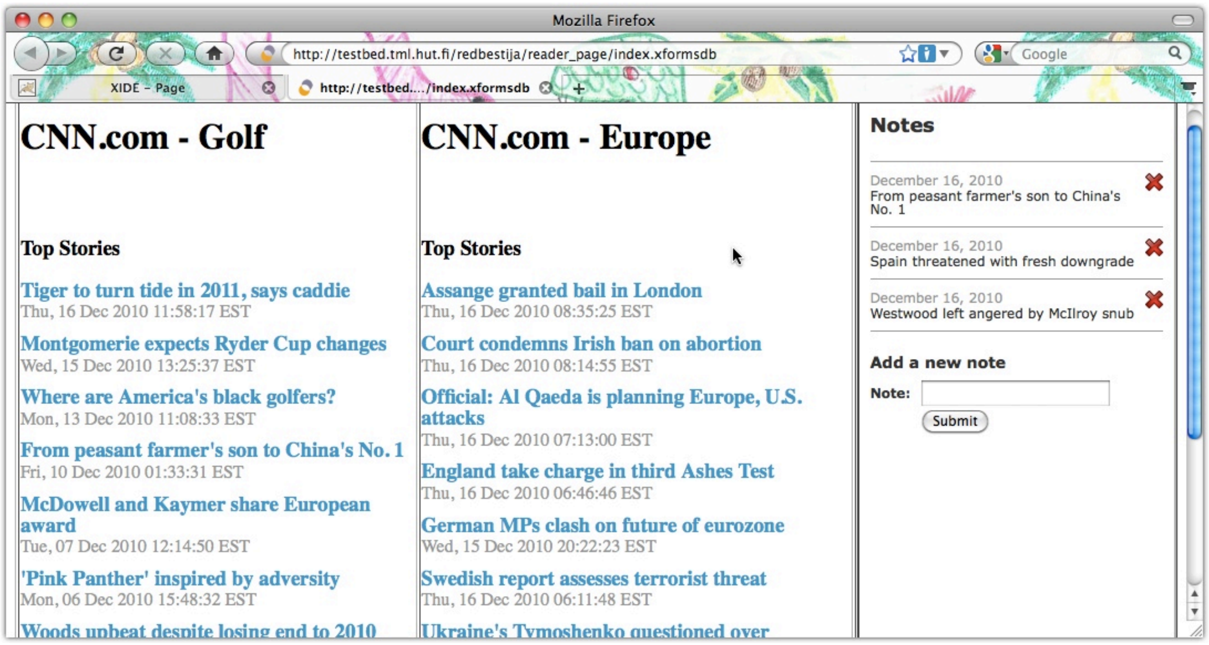
She logs into XIDE and finds her application. She selects the RSS component and changes its CSS file. She uses the Preview tab again in order to see the effect of her changes.

When Alice succeeds in CSS editing, she also decides to modify the functionality of component. She did not like that the RSS reader shows too many news and the their amount cannot be defined. She opens the source code of the component and manages to add a new parameter to the component. This parameter allows to set how many news will be shown.

Contribute new component

Alice decides to share her new advanced RSS reader component with the other people who use XIDE. The sharing procedure requires pressing a single button.

Finally Alice updates the published application with a new version (see Figure 22).


Mozilla Firefox
http://testbed.tml.hut.fi/redbestija/reader_page/index.xformsdb
Google
XIDE - Page
http://testbed..../index.xformsdb

CNN.com - Golf

Top Stories

Tiger to turn tide in 2011, says caddie
Thu, 16 Dec 2010 11:58:17 EST

Montgomerie expects Ryder Cup changes
Wed, 15 Dec 2010 13:25:37 EST

Where are America's black golfers?
Mon, 13 Dec 2010 11:08:33 EST

From peasant farmer's son to China's No. 1
Fri, 10 Dec 2010 01:33:31 EST

McDowell and Kaymer share European award
Tue, 07 Dec 2010 12:14:50 EST

'Pink Panther' inspired by adversity
Mon, 06 Dec 2010 15:48:32 EST

Woods upbeat despite losing end to 2010

CNN.com - Europe

Top Stories

Assange granted bail in London
Thu, 16 Dec 2010 08:35:25 EST

Court condemns Irish ban on abortion
Thu, 16 Dec 2010 08:14:55 EST

Official: Al Qaeda is planning Europe, U.S. attacks
Thu, 16 Dec 2010 07:13:00 EST

England take charge in third Ashes Test
Thu, 16 Dec 2010 06:46:46 EST

German MPs clash on future of eurozone
Wed, 15 Dec 2010 20:22:23 EST

Swedish report assesses terrorist threat
Thu, 16 Dec 2010 06:11:48 EST

Ukraine's Tymoshenko questioned over

Notes

December 16, 2010
From peasant farmer's son to China's No. 1

December 16, 2010
Spain threatened with fresh downgrade

December 16, 2010
Westwood left angered by McIlroy snub

Add a new note

Note:

Submit

Figure 22 XIDE: a sample application

7.1.1. Discussion

As it was defined in requirements, the main goal of XIDE is to support gentle level of complexity and provide a situational support for end users tasks.

Extending the approach of (MacLean, Carter, Lovstrand, & Moran, 1990)(Lieberman, Paterno, Klann, & Wulf, 2006), XIDE provides several levels of modification complexity. Because of the design of Template Language and selection of XFormsDB as a background technology for components, skills gained on the current level form a background to achieve the next level. The effort need to be spent on each upgrade is relatively small because of combining visual customizable components, more suitable background technology and other techniques that helps user in the development activities.

The process of sample application development described above demonstrates different levels of modification provided by XIDE in order to smooth the complexity rise and eliminate learning barriers. The estimated slope of complexity for XIDE is depicted on Figure 23.

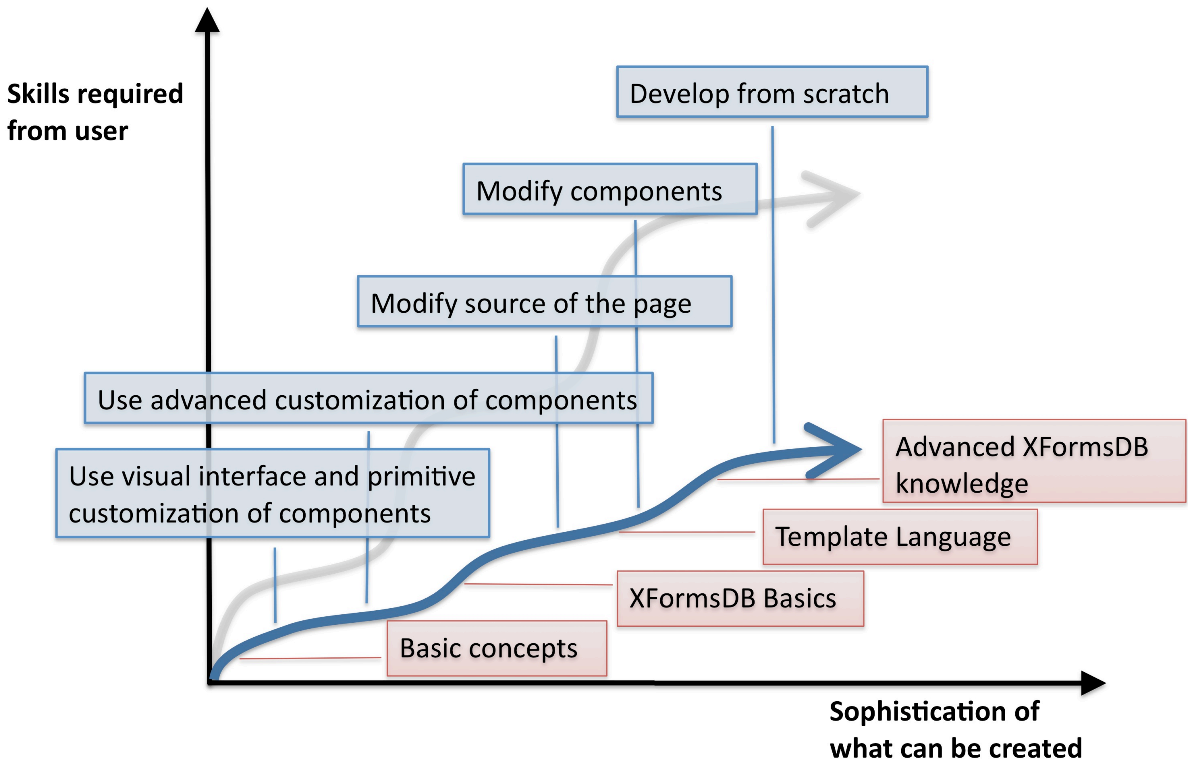

Skills required from user
Develop from scratch
Modify components
Modify source of the page
Use advanced customization of components
Use visual interface and primitive customization of components
Advanced XFormsDB knowledge
Template Language
XFormsDB Basics
Basic concepts
Sophistication of what can be created

Figure 23 Slope of complexity in XIDE

For inexperienced end users it is relatively simple to start using XIDE. It does not require any installation. It allows to create new application without configuring any technical parameters. It provides sample applications and pages and access to existing applications and pages, which can be used as examples or starting point to modify them and implement own functionality.

On the first level of modification end users are supposed to compose pages by visual drag-n-drop of self-sufficient components and customizing by setting parameters. On this level end users can create a Web application without any coding using only visual interfaces, wizards and form-based interfaces. They can view developing application in several different representations, such as design view, tree hierarchy or source code.

When users become familiar with the components and their customization, they can smoothly proceed to the next level. Advanced customization allows end users to configure complex components and link insufficient components with each other. The tool provides wizards to facilitate the process of setting advanced parameter values. Additionally, end users can refer to demo applications to gain the knowledge of making the configuration.

On the next level, user is supposed to make minor modifications in the source code of the page. This activity requires basic knowledge of XHTML and XFormsDB. However, many end users are familiar with markup languages and thus the transition to the source code is not difficult. The tool provides environment that facilitates edition of the source code by means of syntax highlighting and error notifications. XIDE provides a preview mode, where whole functionality of the application is shown. This preview is updated automatically in order to reflect changes in the source code immediately. A Web page can be saved for future reuse as a component without adding any technical details.

Next level requires knowledge of Template Language in order to edit the component. The smooth transition is supported by the fact that TL has transparent and meaningful structure, and it is from the same family as XFormsDB. After modifications to component source code have made, user can easily add new component to the database for future reuse.

On the last stage most sophisticated end users implement the functionality of their application from scratch. However, this task is less complicated than traditional Web application development, because of the XFormsDB technology itself and the evolution of the end user through all previous steps, where some background knowledge was received.

7.2. Usability testing

Initially it was planned to perform a usability testing using paper based design sketches. The main goal of the testing on this stage was to test the main design ideas, such as application and page views, component drag-n-drop, and presentation of properties of the selected element. However, after first trial testing it appeared that the paper mockup itself does not pass the design ideas well. It was designed that the system responds on different user actions (such as mouse hover or mouse movement), but it is not possible to test those issues using paper mockup. So, it was decided to postpone the testing until first interface prototype would be made.

First prototype of XIDE demonstrated further system layout without any underlying functionality. On that stage the prototype consisted of Application view and Page view and provided all UI features, required for performing test tasks.

The goal of the testing was to check the following questions: Is the designed UI usable in general? Do users understand the paradigm of reusable customizable components? How do they feel about drag-n-drop and IDE-like environment? Secondary goal was to check if existing metaphors, terminology and descriptions used in XIDE are understandable for users.

According to the major goal, task list for the testing was designed as a basic system workflow goes. It involves all basic functionality of the XIDE: creating an application, creating a page, looking for a component in the database, adding it to the page, customizing a component, editing source code, saving a page, and publishing an application.

Five participants were planned to participate in the testing. According to Nielson, a known usability expert, this amount of users is enough to show most of usability problems. (Nielsen, 2000) Several authors criticized this assumption, however it is agreed that number of users is not so important as the fact that these users belong to the target group of the system. (Faulkner, 2003) (Spool & Schroeder, 2001) This testing was not supposed to reveal all usability problems of the XIDE tool, but to identify the major flaws. Hence, taking into account that resources of the project did not allow to have usability research with many users, it was decided to stop on five, but select them carefully. Selected participants represented all system target groups. According to personas designed in Section 5.1, there were two programmers, two non-programmers and one expert. They have never used XIDE before. Please refer to the Appendix A for detailed description of the participants and to Appendix B for detailed description of the testing procedure and tasks.

7.2.1. Discussion

The main result of the testing was that all users said they understood the ideas of the UI and thought that the UI is usable. However, the design itself should be improved to be more intuitive for beginners. Users said that it became usable after one understood the idea and learned how to use the system, but it can be difficult to use it for the first time. Most of the users thought that the system should provide more hint information.

Most of users were able to complete all tasks easily except the task, which was related to direct text edition and thus required knowledge of XFormsDB and advanced knowledge of XML. Users said that the task of editing the source code was too complex for them. They were complaining that they did not understand what was going on in source code, what was the syntax and where to add their changes.

Quantitative results showed that task related to direct text edition was difficult for the users to complete and was the most error-prone. Other tasks had low level of errors and were completed in appropriate time.

A lot of comments and ideas were received from the testing. Please refer to the Appendix C to see the most important observations, which influenced on the further redesign. These observations are summarized and grouped by part the UI they related to.

Results of usability testing and recommendations were employed for updating the UI of the XIDE. Special attention was paid to insure that the recommendation improved the interface in general, not only solved concrete problem and also did not initiate new usability lacks. (Molich, Jeffries, & Dumas, 2007)

7.3. Expert evaluation

Besides testing XIDE with real users, it can be examined by experts with some usability inspection method. Inspection methods are used to evaluate the interface against well-known standards. (Holzinger, 2005) Inspections methods require less resources, but produce valuable results if carefully selected, planned and executed. Moreover it can be useful to look on the system from expert point of view. If inspection methods are combined with usability test methods, they can supplement each other and finally provide better understanding of system usability.

In case of XIDE, it was first decided to use *heuristic evaluation* method. Heuristic evaluation assumes that interface is evaluated against list of usability principles. Heuristic evaluation method is criticized for separation from end users, but in case of XIDE it is combined with usability testing, so users are involved in the process. It is really important to select the appropriate heuristic, since it should correspond to the system being evaluated. (Holzinger, 2005)

While looking for suitable heuristic, another method, *Cognitive Dimensions (CDs) of Notations* framework, was found. CDs framework is used for describing and analyzing the usability of system of notations. (Green & Petre, 1996) It was designed specially to evaluate usability of information-based artifacts (or notations); it focuses on aspects of learning and understanding of things. (Green & Blackwell, 1998) CDs. It defines set of dimensions that can be used to describe and evaluate the usability of the existing system. Although CDs framework was initially designed to evaluate visual programming languages, it can be successfully used for evaluation of user interface, where information representation is important. (Dagit, Lawrance, Neumann, Burnett, Metoyer, & Adams, 2006)

Except main focus of the framework, there are other advantages. It is advertised by the authors to be easy to learn and easy to apply. CDs is a lightweight approach for evaluation of UI, while other heuristic evaluations require highly experienced experts to participate and assume more deep and detailed evaluation process. (Green & Blackwell, 1998) (Holzinger, 2005)(Green & Petre, 1996) CDs framework provides qualitative results and even suggests the ways how to overcome the usability problem being found. It can be used for general evaluation to be sure that there is no big problems missed. (Green & Blackwell, 1998)

Authors of CDs defined different user activities, such as incrementation, transcription, modification and exploratory design. Each activity has different set of high-priority dimensions, their relations and possible trade-offs.

For XIDE, Page view and process of Web page creation will be evaluated as a main notation. Modification was chosen as a user activity for evaluation. The following review covers dimensions, which were considered to be most problematic for chosen user activity.

Viscosity: How difficult is it to make minor changes?

Using of components paradigm allows users to make changes to page structure (add, remove, change components order) easily. Also, components are customizable using parameters, so component's behavior or appearance can be changed rapidly, if the components are properly designed. However, if the change being planned is about managing component-to-component or component-to-database relations, it requires more user actions to accomplish the goal.

Hidden dependencies: Can links between important elements be easily visible?

Generally, relations between Web page, containers and components are displayed both graphically and in a tree structure. Also, tabs act as coordinative windows; they support unified highlight notation in order to show, which element is selected and other representations of this element.

During development of more advanced Web pages, there are hidden dependencies between data instances of the parent and child objects. Although on the Data Instance tab user can see which instances are inherited, this issue should be further investigated.

Premature commitment: Is there any decisions, which should be made before all necessary information is available for a user?

Searching for components does not require making any preliminary decisions, since user can easily try the component and remove it if it does not fit.

Using of page samples is one example of premature commitment: if user decides to use a page template, it should be selected before the development starts. However, for professional user it is always possible to change page layout directly according to user needs, by editing its source code.

Abstraction gradient: What is the level of abstraction?

Major XIDE abstractions are obvious and there are just three of them: Web page, containers and components. For non-professional users it is easy to learn and use these. However, for professionals it is possible to change abstraction level and start edit the source code, where there are abstractions like queries, data instances and others inherited from XFormsDB language.

Visibility and juxtaposability: Is it difficult to make an item visible? Is it possible to view two items at the same time and compare them?

There are no objects, which require several steps to be viewed. In XIDE, each abstraction can be easily decomposed into parts, which can be reviewed in the bottom tabs.

There is an advanced search engine to help user to find and view desired component.

Juxtaposability issues are not covered for the source code, queries, etc., however on the higher level of abstraction two components can be placed on design tab to be compared.

Error-proneness: Are there means to predict and avoid mistakes?

Several issues are designed in XIDE to help user make less mistakes. Here are some of these features: mechanism preventing from unsaved data loss, validation of the user data entered in the wizards, fast preview of the page, advanced messages from system, syntax highlight with error checking. However, there is still a room for improvements.

Hard mental operations: Are there hard mental operations required from user?

Abstractions used in XIDE reduce complexity of the perceived system. Other issues make understanding of the page structure easier: design tab displays estimated user interface of the page; navigation tab contains tree representation of the page structure.

Progressive evaluation: is it possible to see the intermediate results?

Preview tab is designed to help user in progressive evaluation: it is possible to see how the deployed page will look like anytime during the page development. Also, design tab helps to estimate the final look during page editing.

7.3.1. Discussion

The goal of this evaluation was not to prove that XIDE is usable. Moreover, CDs cannot be used as means for such kind of proving.(Green & Blackwell, 1998) The evaluation is intended to help to answer the following questions: What are strengths and weaknesses of XIDE in relation to information representation and management? Does design decision made at the beginning of development works? How the tool should be improved?

Finally, evaluation showed no big problems were detected for non-professional users, who use general system workflow described in Section 5.2. XIDE component-based approach and abstractions (components, containers) allow users to manage complex structures easier and there are many features to help user.

There are several features to assist professional users as well, however professional user support should be further investigated to overcome hidden dependencies between different parts of components and pages (e.g., data instances and queries).

8. Conclusions

This Chapter contains conclusions of the Thesis work. It revises research questions and gives brief summarize of the answers. Later, general conclusions of the work related to XIDE system development are introduced. Finally, main contributions and future work are described.

At the beginning of this research, the following research questions were defined:

Question 1: How to combine component-based approach with XFormsDB framework in order to design a tool that supports gentle slope of complexity?

Question 2: What features should the tool provide to facilitate end users development?

Question 3: What are common user interface approaches used in end user development tools?

According to the main research question (Question 1) of this Thesis defined at the beginning, the aim of this research was to combine component-based approach and XFormsDB framework in order to create a tool with gentle slope of complexity. The objective was to answer this question by implementing and evaluating the XIDE tool.

In order to create the tool, the extensive research of end user tasks, challenges, expectations and experience with existing tools reported in the publications was conducted. The result of this review is a set of barriers that end users face during Web development process. The tool that can be used by end users for creating Web applications needs to hide these barriers or assist end user in overcoming them. The requirements, designed for the XIDE tool, describe features that a tool should provide to facilitate end user development. Thus, the XIDE requirements are essentially the answer to the second research question (Question 2).

At the same time with research of end user problems, existing solutions for end user Web development were investigated. Both research and commercial area approaches and tools were studied in order to review the state of the art of end user Web development domain. This review presents the answer to the third research question (Question 3).

Based on requirements and results of state of the art review, a user interface design for XIDE was created. Much attention was paid on selecting appropriate interaction style for the system and creating easy-to-use interface. First paper-based mockups were created and taken through iterative discussion and improving process. Later on, early prototype was evaluated during usability testing round. Analysis of usability testing results showed that in general end users found the approach utilized in XIDE UI usable and they understood the metaphors and concepts of XIDE. The UI of XIDE was improved in order to overcome the problems found during the testing.

Finally all critical requirements were implemented in the XIDE prototype, which was used for evaluation. Current version of XIDE allows to go through all steps of XFormsDB application development: user can create, edit and publish the application.

A sample application use case was designed based on analysis of target end users made in background. This sample application was used to demonstrate how XIDE improves the end user Web development process by providing gentle slope of complexity.

XIDE was also examined with help of Cognitive Dimensions of Notations framework. This usability inspection method allows to evaluate user interface with reference to cognitive and usability aspects. The evaluation did not reveal any major usability problems for end users. However it highlighted some issues to be improved for expert users.

Currently, the XIDE working prototype is running at Aalto University internal server and can be accessed and used by everyone[21]. It is planned to use XIDE in other research projects in order to simplify Web development and, on the other hand, test the tool in real life.

XIDE is published as an open source project at Google Code[22]. More details about implementation and user manuals can be found from there.

Both theoretical and practical contributions of the research described in this Thesis are going to be published as a conference paper. The work on the publication is in progress now. Paper is going to be submitted to International Conference of Web Engineering (ICWE) 2011.

8.1. Main contributions

Constructive research method was used in this Thesis, so the research produced both practical and theoretical contributions.

[21] XIDE working prototype http://testbed.tml.hut.fi/xide/
[22] XIDE open source project page http://code.google.com/p/xformsdb-ide/

Practical contribution of this Thesis is the XIDE tool. The development of XIDE included requirement elicitation, design of XIDE concepts, design of XIDE UI, implementation and evaluation of XIDE. All this tasks, except several technical issues described below, were done by the author of this Thesis.

The work was done in collaboration with other members of Web Services Research Group of Department of Media Technology of Aalto University. The Template Language concept and syntax design was made by Markku Laine with the participation of the author. Implementation of Template Language transformation that was made by Sebastian Monte. Implementation of the components used for XIDE evaluation was done by Markku Laine.

There are several theoretical contributions of this Thesis. First, during background study, problems and expectations of end users who try to develop Web applications were investigated. General classification of end user problems proposed by (Ko, Myers, & Aung, 2004) was employed for a specific case of end user Web development with the focus on problems related to component-based development. The contribution of this research is the classified list of barriers faced by end users during Web development. This research result can be utilized by other researchers who aim to create a Web development tool for end users.

The second contribution is the investigation of the idea of the combining component-based development and XFormsDB framework that was invented in Web Services Research Group. It was proposed that this approach could provide gentler slope of complexity than available systems for end user Web development. While the general idea was invented earlier, the exact way how these two approaches can be combined is proposed in this Thesis and demonstrated by a mean of implementing the XIDE prototype. The evaluation of XIDE showed that the proposed approach of combining a declarative framework, component-based development and best practices from end user research area is a promising solution and should be further investigated.

8.2. Future research

XIDE development process has reached some valuable results and now it is reasonable to offer it to external people for testing. In order to bring XIDE into real world, there are two major issues to be investigated.

First, the component database should be increased to provide various options for a user. A relevant set of components can be taken from a review of publications about typical composites and functionalities of Web application, e.g., (Rode & Rosson, 2003), (Ginige, De Silva, & Ginige, 2005). This set can be verified and increased by analyzing information from extensive survey of possible end users. Initial set of components should be developed by XFormsDB experts; the development can be made using XIDE.

Another issue, which is actually combined with the first one, is piloting XIDE with real users. Is can bring a feedback about the usability of the system; also it can give a lot of hints about what kind of components users really need.

Besides testing XIDE in real life, there are also several things to be added into the system. One of them is to add Web 2.0 features to XIDE, as it was mentioned in XIDE requirements. Web 2.0 features provide options for communication and sharing information, which are highly appreciated by users. (Jazayeri, 2007) According to (McAfee, 2006), there are six components that form Web 2.0 paradigm. XIDE already includes some of them, e.g., search engine for components, tags for components and samples, possibility to contribute a component. However, it can be expanded by adding a possibility to comment a component from the database or subscribe to updates for the component.

There are other features, which are not implemented yet, are:

- Sample pages and applications, which users can utilize and build their own applications by extending them.
- Component versioning, which allows to maintain different versions of one component.
- CSS problem. Currently, users should create their CSS files without ability to use UI element name from the main source file (XFormsDB). This problem is caused by XForms processor, which changes element names. So, user has to know how the element will be called after it will be transformed to HTML in order to be able to assign CSS rule for it.

Bibliography

Barr, P., Biddle, R., & Noble, J. (2002). *A Taxonomy of User-Interface Metaphors*. Victoria University of Wellington, New Zealand, School of Mathematical and Computing Sciences. Citeseer.

Bass, L., Clements, P., & Kazman, R. (2003). *Software architecture in practice*. Addison-Wesley.

Beletski, O. (2008). End User Mashup Programming Environments.

Benyon, D., Turner, P., & Turner, S. (2005). *Designing interactive systems*. Addison-Wesley.

Boyer, J. (2009 20-October). *W3C Standards*. Retrieved 2010 16-December from XForms 1.1 : http://www.w3.org/TR/xforms11

Brandt, J., Guo, P., Lewenstein, J., & Klemmer, S. (2008). Opportunistic programming: how rapid ideation and prototyping occur in practice. *Proceedings of the 4th international workshop on End-user software engineering, IWEUSE* . ACM.

Browser market share, N. A. (2010 15-December). *Browser Market Share*. Retrieved 2010 15-December from Market share for browsers, operating systems and search engines: http://marketshare.hitslink.com/browser-market-share.aspx?qprid=0

Calabria, T. (2004 02-March). *Step Two Designs*. Retrieved 2010 15-December from An introduction to personas and how to create them: http://www.steptwo.com.au/papers/kmc_personas/index.html

Cao, J., Riche, Y., Wiedenbeck, S., Burnett, M., & Grigoreanu, V. (2010). End-user mashup programming: through the design lens. *Proceedings of the 28th international conference on Human factors in computing systems*. ACM.

Clements, P., Garlan, D., Little, R., Nord, R., & Stafford, J. (2003). Documenting software architectures: views and beyond. *Proceedings of the 25th International Conference on Software Engineering*. IEEE Computer Society.

Costabile, M., Fogli, D., Fresta, G., Mussio, P., & Piccinno, A. (2004). Software Environments for End-User Development and Tailoring. *PsychNology Journal* .

Costabile, M., Mussio, P., Parasiliti Provenza, L., & Piccinno, A. (2008). End users as unwitting software developers. *Proceedings of the 4th international workshop on End-user software engineering, WEUSE'04*. ACM.

Crnkovic, G. (2010). Constructive Research and Info-Computational Knowledge Generation. *Model-Based Reasoning in Science and Technology* .

Cypher, A., Lau, T., Nichols, J., & Dontcheva, M. (2009). Workshop on end user programming for the web. CHI '09 Proceedings of the 27th international conference extended abstracts on Human factors in computing systems . ACM.

Dagit, J., Lawrance, J., Neumann, C., Burnett, M., Metoyer, R., & Adams, S. (2006). Using cognitive dimensions: advice from the trenches. *Journal of Visual Languages & Computing , 17* (4), 302--327.

Douglass, R., Little, M., & Smith, J. (2005). Building online communities with Drupal, phpBB, and WordPress. Apress.

Dubinko, M. (2003). *XForms essentials*. O'Reilly.

Ennals, R., & Gay, D. (2007). User-friendly functional programming for web mashups. *Proceedings of the 12th ACM SIGPLAN international conference on Functional programming*. ACM.

Faulkner, L. (2003). Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, & Computers , 35* (3), 379.

Fiala, Z., Hinz, M., Meissner, K., & Wehner, F. (2003). A component-based approach for adaptive, dynamic web documents. *Journal of Web Engineering* .

Floyd, I., Jones, M., Rathi, D., & Twidale, M. (2007). Web Mash-ups and Patchwork Prototyping: User-driven technological innovation with Web 2.0 and Open Source Software. *Proceedings of the 40th Annual Hawaii International Conference on System Sciences*. IEEE Computer Society.

Fraternali, P. (1999). Tools and approaches for developing data-intensive Web applications: a survey. *ACM Computing Surveys* .

Gaedke, M., & Graef, G. (2001). Development and Evolution of Web-Applications Using the WebComposition Process Model. *Web Engineering, Software Engineering and Web Application Development* , 58--76.

Gaedke, M., & Rehse, J. (2000). Supporting compositional reuse in component-based Web engineering. *Proceedings of the 2000 ACM symposium on Applied computing-Volume 2* (p. 933). ACM.

Gaedke, M., Segor, C., & Gellersen, H. (2000). WCML: paving the way for reuse in object-oriented Web engineering. *Proceedings of the 2000 ACM symposium on Applied computing*. ACM.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: elements of reusable object-oriented software*. Addison-wesley Reading.

Garrett, J. (2005). Ajax: A New Approach to Web Applications. *Nine* .

Ginige, A., & Murugesan, S. (2002). Web engineering: An introduction. *Multimedia, IEEE* .

Ginige, J., De Silva, B., & Ginige, A. (2005). Towards end user development of web applications for SMEs: A component based approach. *Web Engineering* .

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

Green, T., & Blackwell, A. (1998). Cognitive dimensions of information artefacts: a tutorial. *BCS HCI Conference*.

Green, T., & Petre, M. (1996). Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. *Journal of Visual Languages and Computing* .

Hartmann, B., Doorley, S., & Klemmer, S. (2008). Hacking, mashing, gluing: Understanding opportunistic design. *Pervasive Computing, IEEE* .

Hartmann, B., Wu, L., Collins, K., & Klemmer, S. (2007). Programming by a sample: Rapidly prototyping web applications with d. mix. *Proceedings of UIST*. Citeseer.

Holzinger, A. (2005). Usability engineering methods for software developers. *Communications of the ACM , 48* (1), 71--74.

Honkala, M. (2006). *Web User Interaction – A Declarative Approach based on XForms*. Helsinki University of Technology, Telecommunications Software and Multimedia Laboratory. Espoo: Otamedia Oy.

Hostetter, M., Kranz, D., Seed, C., Terman, C., & Ward, S. (1997). Curl: A gentle slope language for the web. *World wide web journal* .

HTML 4.01 Specification. (1999 24-December). Retrieved 2010 16-December from Standards W3C: http://www.w3.org/TR/html401/

Hundhausen, C., Farley, S., & Brown, J. (2009). Can direct manipulation lower the barriers to computer programming and promote transfer of training?: An experimental study. *ACM Transactions on Computer-Human Interaction (TOCHI)* .

IEEE Standard 1471-2000, 2. (2000). IEEE-Std-1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems. *IEEE, http://standards.ieee.org*

Järvinen, P. (2004). *On Research Methods*. Tampere: Opinpajan kirja.

Jazayeri, M. (2007). Some trends in web application development. *Future of Software Engineering, 2007. FOSE'07* , 199--213.

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming. *ACM Computing Surveys* .

Kim, M., Bergman, L., Lau, T., & Notkin, D. (2004). An ethnographic study of copy and paste programming practices in OOPL. *Proceedings od 2004 International Symposium on Empirical Software Engineering, 2004. ISESE'04*. IEEE.

Klann, M., Paterno, F., & Wulf, V. (2006). Future Perspectives in End-User Development. *End User Development* .

Ko, A., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., et al. (2009). The State of the Art in End-User Software Engineering. *Journal ACM Computing Surveys* .

Ko, A., Myers, B., & Aung, H. (2004). Six learning barriers in end-user programming systems. *2004 IEEE Symposium on Visual Languages and Human Centric Computing*. IEEE.

Kuniavsky, M. (2003). *Observing the user experience : a practitioner's guide to user research*. San Francisco , (Calif.), USA: Morgan Kaufmann Publishers.

Kuuskeri, J., & Mikkonen, T. (2009). Partitioning web applications between the server and the client. *Proceedings of the 2009 ACM symposium on Applied Computing*. ACM.

Laine, M. P. (2010). XFormsDB—An XForms-Based Framework for Simplifying Web Application Development. Helsinki: M.Sc. Thesis.

Leff, A., & Rayfield, J. (2001). Web-Application Development Using the Model/View/Controller Design Pattern. *Proceedings of the 5th IEEE International Conference on Enterprise Distributed Object Computing*. IEEE Computer Society.

Lieberman, H. (2001). Your wish is my command: Programming by example. Morgan Kaufmann.

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

Long, F. (May 2009). *Real or Imaginary; The effectiveness of using personas in product design*. Dublin: Proceedings of the Irish Ergonomics Society Annual Conference 2009.

MacLean, A., Carter, K., Lovstrand, L., & Moran, T. (1990). User-tailorable systems: pressing the issues with buttons. *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*. ACM.

Malhotra, A., Melton, J., & Walsh, N. K. (2010 14-December). *XQuery 1.0 and XPath 2.0 Functions and Operators (Second Edition)*. Retrieved 2010 16-December from W3C Standards: http://www.w3.org/TR/xpath-functions/

Mark, R. L., & Nielsen, J. (1994). *Usability inspection methods*. Wiley New York.

Maximilien, E., Ranabahu, A., & Gomadam, K. (2008). An Online Platform for Web APIs and Service Mashups. *IEEE Internet Computing* , 32--43.

McAfee, A. (2006). Enterprise 2.0: The dawn of emergent collaboration. *MIT Sloan Management Review* .

McCreary, D. (2007 14-December). *Introducing the XRX Architecture: XForms/REST/XQuery*. Retrieved 2010 14-December from Dr. Data Dictionary : http://datadictionary.blogspot.com/2007/12/introducing-xrx-architecture.html

Molich, R., Jeffries, R., & Dumas, J. (2007). Making usability recommendations useful and usable. *Journal of Usability Studies* , 162--179.

Morch, A., Stevens, G., Won, M., Klann, M., Dittrich, Y., & Wulf, V. (2004). Component-based technologies for end-user development. *Communications of the ACM* .

Myers, B. (March 1998). *A brief history of Human Computer Interaction* (Vol. 2). ACM interactions.

Myers, B., & Ko, A. (2009). The past, present and future of programming in HCI. *Human-Computer Interaction Consortium (HCIC'09)* .

Myers, B., Hudson, S., & Pausch, R. (2000). Past, present, and future of user interface software tools. *ACM Transactions on Computer-Human Interaction (TOCHI)* .

Myers, B., Ko, A., & Burnett, M. (2006). Invited research overview: end-user programming. *CHI'06 extended abstracts on Human factors in computing systems*. ACM.

Myers, B., Pane, J., & Ko, A. (2004). Natural programming languages and environments. *Communications of the ACM* .

Nardi, B. (1993). A small matter of programming: perspectives on end user computing. The MIT Press.

Neale, D., & Carroll, J. (1997). The role of metaphors in user interface design. *Handbook of human-computer interaction* , 441--462.

Newman, M., Lin, J., Hong, J., & Landay, J. (2003). DENIM: An informal web site design tool inspired by observations of practice. *Human-Computer Interaction* .

Nielsen, J. (2000 19-March). *Usability Testing with five users*. Retrieved 2010 14-December from Jakob Nielsen's Website: http://www.useit.com/alertbox/20000319.html

North, C., & Shneiderman, B. *A Taxonomy of Multiple Window Coordinations*. University of Maryland, Human-Computer Interaction Laboratory. 1997: Citeseer.

(2009). Open Source CMS Market Share Report 2009. water&stone and CMSWire.

Pane, J., & Myers, B. (1996). *Usability issues in the design of novice programming systems*. Carnegie Mellon University, School of Computer Science. Pittsburgh: Citeseer.

Pohja, M., Honkala, M., Penttinen, M., Vuorimaa, P., & Ervamaa, P. (2007). Web User Interaction. *Web Information Systems and Technologies* .

ProgrammableWeb - Mashups, APIs, and the Web as Platform. (2010 14-December). Retrieved 2010 14-December from www.programmableweb.com

Ramirez, F., & Wroblewski, L. (2005 10-April). *Web Application Solutions: A Designer's Guide*. Retrieved 2010 15-December from LukeW Ideation + Design: http://www.lukew.com/ff/entry.asp?170

Repenning, A., & Ioannidou, A. (2006). What makes end-user development tick? 13 design guidelines. *End User Development* .

Rode, J., & Rosson, M. (2003). Programming at runtime: requirements and paradigms for nonprogrammer web application development. *Proceedings of the 2003 IEEE Symposium on Human Centric Computing Languages and Environments*. IEEE Computer Society.

Rode, J., Bhardwaj, Y., Perez-Quinones, M., Rosson, M., & Howarth, J. (2005). As easy as “Click”: End-user web engineering. *Web Engineering* .

Rode, J., Rosson, M., & Perez-Quinones, M. (2004). End-Users' Mental Models of Concepts Critical to Web Application Development. *2004 IEEE Symposium on Visual Languages and Human Centric Computing*. IEEE.

Rode, J., Rosson, M., & Quinones, M. (2006). End user development of web applications. *End User Development* .

Rosson, M., Ballin, J., & Nash, H. (2004). Everyday programming: Challenges and opportunities for informal web development. *IEEE Symposium on Visual Languages Human-Centric Computing*.

Rosson, M., Ballin, J., & Rode, J. (2005). Who, what, and how: A survey of informal and professional web developers. *2005 IEEE Symposium on Visual Languages and Human-Centric Computing*.

Rothermel, K., Cook, C., Burnett, M., Schonfeld, J., Green, T., & Rothermel, G. (2000). WYSIWYT testing in the spreadsheet paradigm: An empirical evaluation. *Proceedings of the 22nd international conference on Software engineering*. IEEE.

Sano, D. (1996). Designing large-scale Web sites: a visual design methodology. John Wiley & Sons, Inc.

Scaffidi, C., Shaw, M., & Myers, B. (2005). Estimating the numbers of end users and end user programmers. *2005 IEEE Symposium on Visual Languages and Human-Centric Computing*. IEEE.

Scollan, B., Byrnes, A., Nagle, M., Coyle, P., York, C., & Ingram, M. (2008). *Drupal Usability Research Report* . University of Baltimore, Interaction Design & Information Architecture.

Shaw, M. (2003). Writing good software engineering research papers. *Proceedings of the 25th International Conference of Software Engineering (ICSE'03)*. IEEE Computer Society.

Shneiderman, B. (1983). Direct Manipulation. A Step Beyond Programming Languages. *IEEE Computer* .

Shneiderman, B. (2003). Promoting universal usability with multi-layer interface design. *Proceedings of the 2003 conference on Universal usability* (p. 8). ACM.

Shneiderman, B., & Plaisant, C. (2010). Designing the User Interface: Strategies for Effective Human-Computer Interaction (Fifth Edition). Pearson/Addison Wesley.

Spool, J., & Schroeder, W. (2001). Testing web sites: five users is nowhere near enough. *CHI'01 extended abstracts on Human factors in computing systems* (p. 286). ACM.

Tullis, T. (2005). Web-based Presentation of Information: The Top Ten Mistakes and Why They Are Mistakes. *HCI International 2005 Conference*.

W3Counter, A. W. (2010 November). *Global Web Stats*. Retrieved 2010 December from W3Counter - Live Web Stats and Free Web Counter: http://www.w3counter.com/globalstats.php

Won, M., Stiemerling, O., & Wulf, V. (2006). Component-based approaches to tailorable systems. *End User Development* .

Yang, J., & Papazoglou, M. (2006). Web component: A substrate for web service reuse and composition. *Advanced Information Systems Engineering*. Springer.

Ye, Y., & Fischer, G. (2007). Designing for Participation in Socio-Technical Software Systems. *Proceedings of 12th International Conference on Human-Computer Interaction*. Springer.

Yu, J., Benatallah, B., Casati, F., & Daniel, F. (2008). Understanding mashup development. *IEEE Internet Computing* , 44-52.

Zang, N., Rosson, M., & Nasser, V. (2008). Mashups: who? what? why? *CHI'08 extended abstracts on Human factors in computing systems*. ACM.

Appendix A: Description of usability testing participants

User 1: programmer

He has been working as a software developer for several years and has a M.Sc. in a computer science area. His work is not related to the Web application creation but he did several Web pages in his past just to try the technology (HTML/CSS/JavaScript/PHP). He is definitely and XML/HTML expert user and uses Internet a lot. He has never heard about XFormsDB technology and tried any of mashup tools.

User 2: programmer

He has been working as a Web developer for several years and has a M.Sc. in a computer science area. He does enterprise Web applications (Java EE, HTML, CSS). He is definitely and XML/HTML expert user and uses Internet a lot. He has never heard about XFormsDB technology and tried any of mashup tools.

User 3: non-programmer

She has totally humanitarian education and she has never developed any program or Web site. She heard about XML/HTML, but she has no real knowledge about these technologies. She uses computer in daily life a lot (simple office tools) and surfs though Internet a lot. She has never heard about XFormsDB technology and tried any of mashup tools.

User 4: non-programmer

She is going her work in the statistic area. She has never worked as software developer, however she has some experience gained while she was studying. She made several Web sites for the courses. She has tried XML and HTML somewhere in the past, but didn't feel herself as a professional. She uses computer and Internet a lot. She has never heard about XFormsDB technology and tried any of mashup tools.

User 5: expert

He is going to finish his M.Sc. in computer science area. He is now working as a summer trainee and developing Web applications on XFormsDB technology. He has an experience in different programming and Web sites creation (HTML, PHP), and he is using Internet a lot.

He knows a lot about XFormsDB technology and develops applications using it. He has used mashup tools (i-Google).

Appendix B: Usability testing procedure

First, participants were given a short description of the XIDE, saying what is the main purpose of the system and what parts does it consist of. Then each of them was asked some questions about their experience in Web technologies and Internet. Then they were shown the XIDE tool and asked to perform the following tasks:

1. Create a test application with a page.
2. Edit the page you have created and make a simple query page. It should contain two non-fake questions.
3. Edit the page you have created and add a welcome text on the top (e.g., “Please answer the following questions”).
4. Publish the application you have created.

After task one and task two they were asked to go through the interface and explain what do they think that is and what is a purpose of this part. While each participant was performing the task he was asked probe questions about the things he/she was doing. Finally they were asked to make a short conclusion about the system and asked small questionnaire about what did think about different system parts.

Appendix C: Usability testing observations

1. **General observations:**
   1.1. Most of users used browser back and forward buttons and sometimes lost information
   1.2. Most of users said they would like to have more hints on different buttons and icons in order to check if this button did what they thought.
   1.3. There were a lot of comments about how different text messages and labels are formulated
2. **Application View**
   2.1. Sometimes users were not able to understand that the application tree on the left is a tree and it can be expandable. Root element confused all users
   2.2. Details table was definitely confusing. Users complained they cannot understand that it is a table (because of the design); they tried to edit the entities in the table; they did no see any link between selected element in the tree and table content; Bottom tab was not used as well
   2.3. Users complained that Application view is too complicated; different tabs are not linked with each other; it is not clear where and how to edit information. They became frustrated when tried to do something in this view.
3. **Wizards**
   3.1. Some users complained they do not understand what some of the fields are, since the name of the field is short and sometimes not descriptive
   3.2. “Check URL” button confused users; they complain about lack of automatic check
4. **Page view: using of components**
   4.1. Users succeeded with drag-n-drop paradigm used for managing the components. However, there were several minor details: users tried to drag components without selecting them; some users dropped component too early.
   4.2. Link between component and Bottom tab is not obvious; everybody tried to click or double click on component when they wanted to edit it
   4.3. Parameters were found, however some of users said they would prefer parameters on a separate tab. Other tabs were not accessed.
   4.4. Mostly all users expected that changes should appear right after they made changes.
5. **Page view: editing the code**

5.1. Everybody tried to select a page in the design view using a mouse.
5.2. Only two of five users got that they need to edit a source code. Most of them said they would prefer a special component with a text.
5.3. Users complained the XML source code was a mess, there were a lot of unstructured text and they couldn’t understand where and what they should edit