UDC 519.8, DOI:10.2298/CSIS090422005S

Prompt System Redesign: Shifting to Open Source Technology to Satisfy User Requirements

Igor Svetel[1], Aleksandar Đurović[2], and Vencislav Grabulov[3]

[1] Innovation Center, Faculty of Mechanical Engineering, Kraljice Marije 16, Belgrade, Serbia
isvetel@mas.bg.ac.rs
[2] Society for Structural Integrity and Life DIVK, Belgrade, Serbia
aca@divk.org.rs
[3] Institute for Materials Testing - Institute IMS, Bulevar vojvode Mišića 43, Belgrade, Serbia
venciaiv@eunet.rs

Abstract. The paper describes a redesign project undertaken in a short period to adapt a software system to user needs. Additional goals of the project included a shift to Open Source software and the selection of technology to enable sustainable system development. The paper chronologically describes all phases of the project and provides reasons for all decisions taken during the development process. The paper concludes with a discussion of the merits of the redesign methodology.

Keywords: System redesign; Open Source; Welder’s Passport; JEE; EJB; JSF.

1. Introduction

System redesign is the process of changing the structure of a system while preserving its external behavior. In the field of software development this process is referred as ‘model refactoring’ [1] as distinct from the term ‘refactoring’ [2], which focuses on the modification of software code. The term ‘redesign’ is used in this paper to emphasize the architectural point of view and to highlight dilemmas and decisions that go beyond software construction.

The main intention of the redesign task was the development of an agile or “change resistant” system [3]. The experience with the first system demonstrated that users are not able to clearly identify their requirements before they start to use the software. Further, the rate of change in software technology is so rapid that no one can expect that a system will hold to the same technology for its lifetime.

Since the time frame for the project was short, it was necessary to find a method that would allow the reuse of existing components. This goal has a major impact on achieving deadlines since it enables the adoption of a sustainable design methodology. The redesign philosophy was organized around the notion that both the design process and the resulting software should be amenable to repeatable change cycles and yet enable the reuse of existing components. One way to achieve these requirements is strict adherence to standards and rejection of all non-standard extensions to the technologies.

The sections in the paper are organized in a sequence that reflects the chronological progression of the redesign process as accurately as possible. Yet, the linear nature of the exposition prevents illustration of many decision loops that were part of the process.

2. Former System

Welder's Passport (WP) 1.0 was a commercial implementation of the concept demonstration software developed as part of the E!2774 project. The EUREKA E!2774 project involved numerous institutions from European countries. The application domain was a computer information system that would enable the monitoring of a welder's career from the beginning of his/her training, through gaining certification and during his/her professional carrier until the end of welding activities. The main result of the project was a data model that encompasses all necessary information. The model gained endorsement from all parties in the international welding industry [4].

The WP 1.0 system inherited Oracle as the DBMS from the demonstration software. The choice of all other technologies was driven by the decision to use JDeveloper as the integrated development environment (IDE). Struts was the main development paradigm supported in JDeveloper at the time of the implementation. All other technologies that were applied in the project were those provided as default IDE settings: Oracle ADF, OC4J, and JSP. CSS and JavaScript technologies were included in the final development phase, as the ad hoc solution to the client's need for a more flexible user interface [5].

The E!2774 project supported only the certification scheme approved by the European Federation for welding, joining and cutting (EWF) [6]. Since in Serbia and the surrounding region various certification schemes exist in parallel, it was necessary to include additional data that would enable management of other certification schemes in addition to the approved certification scheme.

The system consists of three separate web interfaces connected to a central database (Fig. 1). The Company interface enables management of the information regarding welders, welding engineers and their certificates. The ATB interface is dedicated to accredited training bodies and enables them to manage information regarding training courses. The ANB interface is designed for accredited national bodies and enables management of the certification process that a particular ANB supervises.

The main criticism was that the WP 1.0 system was driven by the futuristic vision of the joined multinational market in which welders freely move from one project to another. The reality of the market is diverse. Companies that employ welders usually treat their personnel and their knowledge as the company's competitive advantage. They are frightened by the notion that data about welders and their certificates are kept on some computer server located outside the premises of the company, no matter how much the security of the underlying software system is promised. A second criticism voiced by all organizations was that the whole certification process was an unnecessary system feature. Everyone wanted a system that would provide an advantage for their company, and the concept that a system's functionality should depend on all stakeholders was inconceivable. A third criticism was the high cost of the Oracle (database management system (DBMS)) license that was considered prohibitive by companies that insisted on implementing the system in their premises.

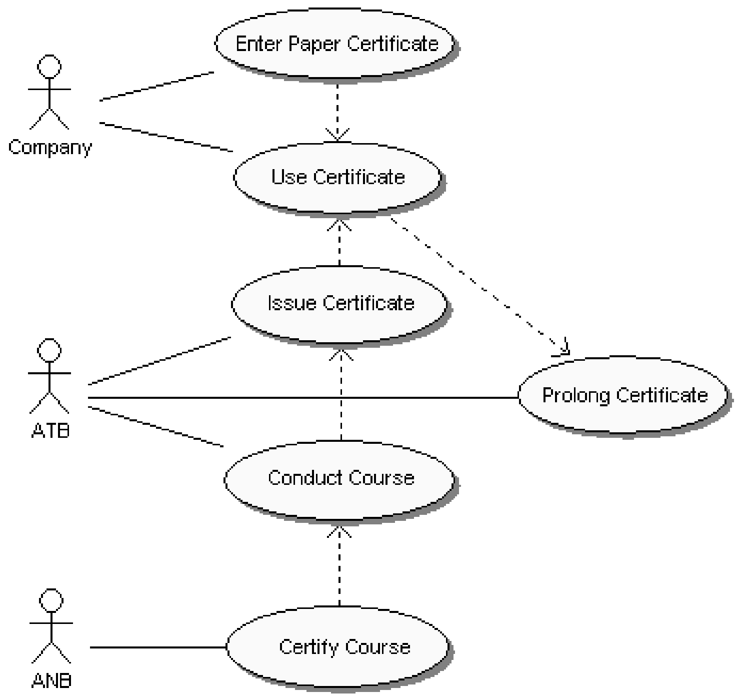


Fig. 1. Structure of the certification scheme

3. System Redesign

The main intent of the system redesign was the inclusion of the system architect in the development team. The development team responsible for the

WP 1.0 system included programmers and welding specialists. Unfortunately, they did not undertake any market research, but based their decisions on the expectations that a maximum level of improvement and change would be welcomed by all users and that technology is a remedy for all problems. The failure of the final product in the market puzzled them because of the positive reactions to their E!2774 research results.

The first phase in the system redesign focused on changing the expectations and habits of all team members. Welding specialists were fond of the certification scheme implementation. It was necessary to conduct a few specially prepared WP 1.0 demonstrations to potential users to persuade welding specialists that the certification scheme implementation was an unnecessary feature. When asked about feature that was of greatest interest to them, the potential users pointed to the process of certification management (i.e., creating, editing, and using). Even ATBs and ANBs perceived certification management as the best feature of the system. All other aspects were seen either as unnecessary or as a potential threat to their business. The programmers had a different set of prejudices. They were proud of their achievement in implementing the complex structure of the certification scheme, and thought that the users should be willing to adapt their usual way of working to the software. Accordingly, they perceived the rejection of the certification scheme as an insult, and it took a great deal of effort to persuade them that the design of a system that is more responsive to user needs will create enough programming challenges.

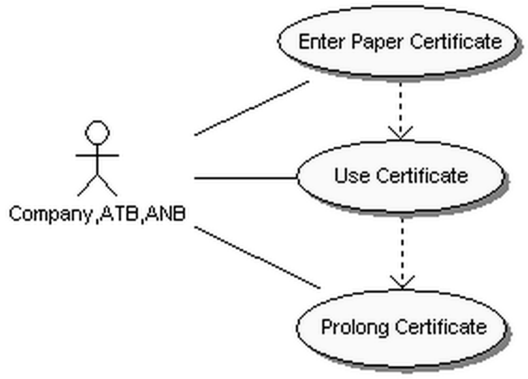


Fig. 2. Structure of the WP 2.0 system

Once the development team was persuaded, the design of WP 2.0 system was outlined. The decision was made that only the certification management process would be implemented as the application running on the stand-alone computer (Fig. 2). In addition it was decided that the full scope of the WP 1.0 database would be implemented, thus preserving compatibility with all of the certification schemes. Also, instead of a single computer application, the full client server technology would be implemented, although hidden from the user. These decisions were taken to enable the evolution of the system as the user needs increase. It is easy to foresee that once a system has been accepted a need for more workstations will quickly surface and, in the not so distant future, there will be an increasing desire by users to access the database from different locations. By implementing a client-server architecture in the first place, the system will easily satisfy all future expansion needs. Also, by preserving a full database structure, the system will support certification schemes once such a need arises.

In this way the development methodology shifted from a strictly sequential model to an iterative model [7]. WP 2.0 was conceived from the start as an ever-changing system. Some changes were easy to predict, like the expansion of the scope of the system and the inevitable change of the underlying technologies. Others, which will occur during the system's lifetime cannot be predicted in advance. Under these circumstances the only solution is to design an adaptive system and adopt an adaptive software development methodology [8]. Accordingly, the term 'change' became a high priority criterion in the whole decision-making process of WP 2.0. All considerations made during system development were finally evaluated in regard to their ability to adapt to the change, and only those solutions that were scored by all development team members as most adaptable were accepted.

4. Selection of the Technology

In answer to the high cost of software licenses, a shift to Open Source software was proposed. At first sight this appeared to be an easy task since all necessary replacement technologies existed as Open Source software. However, this initial optimism quickly vanished as we were faced with the selection of the most appropriate software from the vast amount of available technologies. It soon became obvious that the selection of the proper technologies would be crucial for the success of the redesign project. Since the project had only three months to deliver the new system, it was decided that two months should be allocated for the selection of the technology, since all of the team members were new to the domain of Open Source software.

The selection of the DBMS was based on the positive experience that the whole Open Source community appeared to have had with the MySQL DBMS. Since the maximum size of the MySQL database was not a limiting factor and the support for Unicode existed, it was decided to use this DBMS technology without considering other options. Since MySQL is mature and well-supported technology and is used by many development teams, it was estimated that it would facilitate the software system to be adapted to all future changes.

The decisions on the programming language and the platform for server programming was also straightforward. The same technologies as in WP 1.0

were selected, namely Java and Java Enterprise Edition (JEE). Both Java and JEE are well supported and are clearly among the most adaptable technologies on the market.

The next technology that was taken into consideration was IDE. Although programmers opted for JDeveloper, an environment with which they where already familiar, it was precisely this familiarity that led to the rejection of this technology. In the previous project, the team had been accustomed to adopt any Oracle technology that the JDeveloper system provided as a default. The rejection of JDeveloper was clear sign to them that the new system should not have any resemblance to the old one. In addition, the programmers admitted that every time a new version of JDeveloper arrived they had many problems to migrate the existing version of the system to the new software version. After this decision was taken, two IDEs were considered: Eclipse and NetBeans. Both were seen to provide the same functionality as JDeveloper, and they both seemed to have the same level of adaptability based on the number of supporting companies. The final decision, in favor of Eclipse, was taken much later after the persistence technology had been chosen, since Eclipse had slightly better support for EJB 3.0.

The programmers' tendency to include, somewhat unsystematically, all kinds of technologies in their IDE required some preventative measures. The system architect had a separate development system on which only selected technologies were installed, and used that system to regularly test if he could replicate the programmers results. This precaution prevented programmers from finding easier paths and applying methods that differed from accepted standards. During the development of the system an inconsistency was detected only once, when programmers made their own decision on the inclusion of the technology for PDF creation.

The selection of version control technology was also undertaken with a view to controlling the activities of the programmers. In the WP 1.0 system the programmers had attempted to use CVS technology but had found some inexplicable errors that probably reflected their unwillingness to be controlled. For that reason, the Subversion version control technology was selected without consultation of the programmers. The rationale behind this selection was that it is a new technology developed to overcome the shortcomings inherent in CVS. That fact forestalled any arguments about errors in CVS. At the time of selection Subversion was not among the default version control technologies used in any IDE, but the strength of supporting companies promised its adaptability. This proved to be good forecast since today Subversion is included as a default version control system in all major IDEs. Once Subversion was introduced as the version control technology, a VisualSVN Server 1.1 [9] was installed as a central repository and Subversive [10] was selected as the Eclipse client. Subversion proved itself as a robust technology and the programmers accepted it and were pleased with its automatic merging features. Only once did a problem with synchronization occur, when a programmer took a copy of the whole system, moved it to his home computer, and returned with a modified version. This situation served as a good example that strict adherence to specified technologies is the single most important means to achieve good results.

Because we decided to use a traditional relational DBMS for data storage in combination with Java-based object-oriented technology to access data, the issue of selecting an object-relational mapping technology became an important topic. Two technologies where considered, Hibernate 3.0 and Enterprise JavaBeans 3.0 (EJB 3.0). EJB was the first attempt to provide a standard solution to problems of persistence, transactional integrity, and security. Regardless of the fact that IBM and Sun Microsystems backed this technology, it was perceived to be too complex. Hibernate is one among several technologies developed as an answer to the shortcomings of earlier EJB versions. The technology gained widespread acceptance, and for some time was considered as the best persistence technology. As part of the EJB 3.0 development (JSR 220), the Java Persistence API (JPA) standard was developed as the merger of expertise from TopLink, Hibernate, JDO, and EJB developers. Since JPA accumulates the best properties of all previous persistence technologies and is supported by largest software companies, we decided to select EJB 3.0 as the persistence technology for our project. TopLink Essential [11] was selected as the persistence provider (JPA implementation) because it was the EJB 3.0 reference implementation.

Selection of the web application framework took most of the time allocated for the technology selection process. The number of available technologies is large and any one technology can have several implementations offering different functionality, thus making the selection a challenging task. To reduce complexity we sorted out only those frameworks that are based on the Java programming language. From those, based on popularity in the developer community, we considered the following frameworks: JavaServer Faces, JBoss Seam, Apache Struts, Spring Framework, and Tapestry. After a first round of consideration, Spring and Tapestry were rejected because their compatibility with already selected technologies was not straightforward. Finally, JavaServer Faces was selected as the technology that offers just enough necessary functionality and does not goes into unnecessary extensions. In addition, it offers the cleanest integration with the set of already selected technologies. Once a technology was selected, we had to decide on a particular implementation. This task was particularly difficult because each implementer typically adds their particular component libraries in addition to the basic ones. Fortunately, a live demonstration for each technology can be found on the Internet, and the required functionality can be compared with the actual one. To accomplish this task we used paper and pencil to draw the required user interface. Then, using live demonstrations, we investigated whether a particular implementation could render all necessary components. We finished with two technologies ICEfaces [12] and RichFaces [13] as providing all of the necessary components. The final decision was based on the development of prototype WP systems using both technologies. Both of the prototype systems provided all of the necessary functionality, with similar ease of use. Finally, a slight delay in supporting the latest version of the development technology offered by ICEfaces decided in favor of RichFaces.

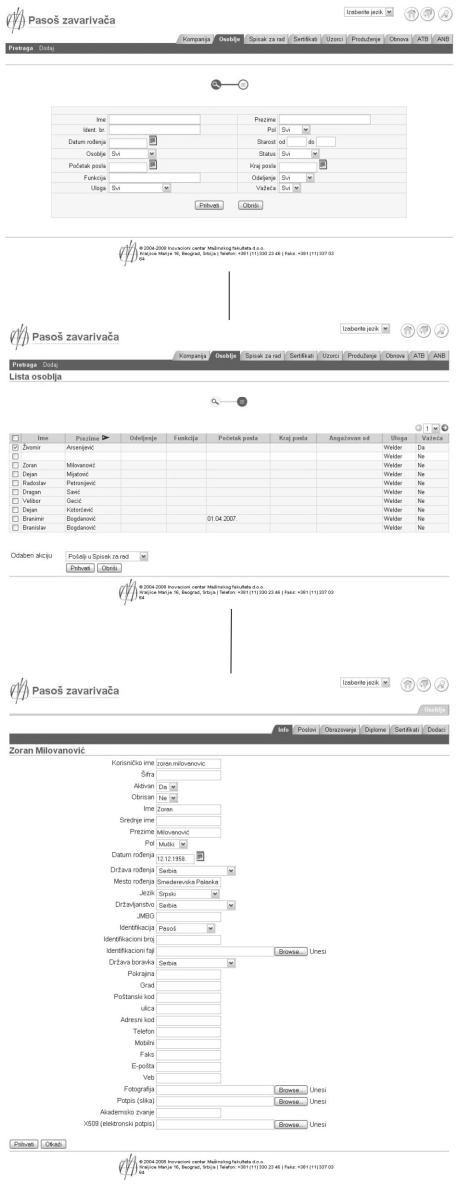
The final decision on the application server took into account all of the selected technologies. The GlassFish server [14] (GlassFish Project - V2 UR2 Final Build) was seen to provide the appropriate combination of support for the selected technologies and appeared to guarantee that its technology would accommodate future changes.

The complete technology selection process took almost two thirds of the time allocated for the project and proved to be a most challenging task. The Open Source community offers many alternative technologies that cover the same functionality and services. To accomplish the selection task, we relied on three types of criteria: a) whether the technology provides all of the required functionality, b) the degree of market acceptance, and c) market strength of the company supporting the particular technology. Those criteria combined with different levels of technology research (information found on the Internet, technical documents, demonstration versions) proved to be sufficient in all cases.

5. GUI Redesign

The WP 1.0 system used a default graphical user interface (GUI) provided by JDeveloper. It was based on tables, bitmap tabs and buttons. The developers did not consider ease of use and system look and feel as important features. Instead, they placed all of their efforts on functionality, with the belief that this is what the market is waiting for. After the first users struggled to enter their data, it became obvious that a more flexible GUI was needed. The user interface in WP 1.0 supported synchronous loading of content. This was often a frustrating experience for the user, because incorrectly entered data were detected only after all of the data had been entered and an attempt was made to submit the data to the database. In addition, a system or communication error during a data entry process required all of the data to be entered again from the beginning.

When the final decision to develop the WP 2.0 system was made, the need for a new GUI was considered as important a feature as the transition to Open Source technologies. By that time it had become obvious that to gain the approval of the users a Web-based application would need to look and feel as any other OS-based computer application. The main GUI change was rejection of the single level tab-based interface (Fig. 3). Instead, a visually recognizable, hierarchical interface based on windows and dialog boxes was adopted (Fig. 4). This design provides clear navigation. At any point in time, the users know on which type of data they are working. This is important because the user needs to consult many external documents to gather data required for entry into the database. During that process, the users are likely to turn away from the computer screen, and sometimes even need to leave their workstation. It is important for maintaining productivity that the users have a clear understanding of the data elements on which they are working upon returning their attention to the computer screen. The selected JSF technology and RichFaces implementation enabled us to meet that requirement.



Ime		Prezime	
Ident. br.		Pol	Svi
Datum rođenja		Starost	od do
Osoblje	Svi	Status	Svi
Početak posla		Kraj posla	
Funkcija		Odeljenje	Svi
Uloga	Svi	Važeća	Svi

© 2004-2008 Inovacioni centar Mašinskog fakulteta d.o.o.
Kraljice Marije 16, Beograd, Srbija | Telefon: +381 (11) 330 23 46 | Faks: +381 (11) 337 03 64

Lista osoblja

	Ime	Prezime	Odeljenje	Funkcija	Početak posla	Kraj posla	Angažovan od	Uloga	Važeća
✓	Živomir	Arsenijević						Welder	Da
								Welder	Ne
	Zoran	Milovanović						Welder	Ne
	Dejan	Mijatović						Welder	Ne
	Radoslav	Petronijević						Welder	Ne
	Dragan	Savić						Welder	Ne
	Velibor	Gecić						Welder	Ne
	Dejan	Kotorčević						Welder	Ne
	Branimir	Bogdanović			01.04.2007.			Welder	Ne
	Branislav	Bogdanović						Welder	Ne

Odaberi akciju: Pošalji u Spisak za rad

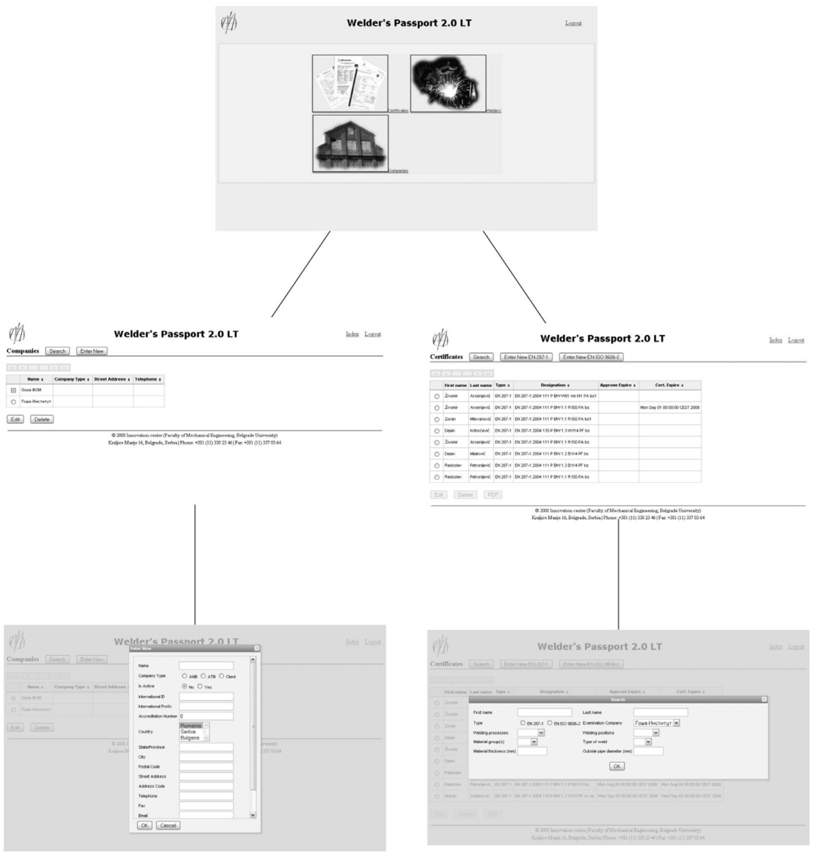
© 2004-2008 Inovacioni centar Mašinskog fakulteta d.o.o.
Kraljice Marije 16, Beograd, Srbija | Telefon: +381 (11) 330 23 46 | Faks: +381 (11) 337 03 64

Zoran Milovanović

Korisničko ime	zoran.milovanovic
Šifra	
Aktivan	Da
Obrisan	Ne
Ime	Zoran
Srednje ime	
Prezime	Milovanović
Pol	Muški
Datum rođenja	12.12.1958.
Država rođenja	Serbia
Mesto rođenja	Smederevska Palanka
Jezik	Srpski
Državljanstvo	Serbia
JMBG	
Identifikacija	Pasoš
Identifikacioni broj	
Identifikacioni fajl	
Država boravka	Serbia
Pokrajina	
Grad	
Poštanski kod	
ulica	
Adresni kod	
Telefon	
Mobilni	
Faks	
E-pošta	
Veb	
Fotografija	
Potpis (slika)	
Akademsko zvanje	
X509 (elektronski potpis)	

© 2004-2008 Inovacioni centar Mašinskog fakulteta d.o.o.
Kraljice Marije 16, Beograd, Srbija | Telefon: +381 (11) 330 23 46 | Faks: +381 (11) 337 03 64

Fig. 3. Visually linear nature of the WP 1.0 interface



Companies

	Name	Company Type	Street Address	Telephone
●	Gosa ФОМ			
○	Гоша Институт			

© 2008 Innovation center (Faculty of Mechanical Engineering, Belgrade University)
Kraljice Marije 16, Belgrade, Serbia | Phone: +381 (11) 330 23 46 | Fax: +381 (11) 337 03 64

Certificates

	First name	Last name	Type	Designation	Approve Expire	Cert. Expire
○	Živomir	Arsenijević	EN 287-1	EN 287-1:2004 111 P BW W01 nm t41 PA bs1		
○	Živomir	Arsenijević	EN 287-1	EN 287-1:2004 111 P BW 1.1 R t50 PA bs		Mon Sep 01 00:00:00 CEST 2008
○	Zoran	Milovanović	EN 287-1	EN 287-1:2004 111 P BW 1.1 R t50 PA bs1		
○	Dejan	Kotorčević	EN 287-1	EN 287-1:2004 135 P BW 1.3 W t14 PF bs		
○	Živomir	Arsenijević	EN 287-1	EN 287-1:2004 111 P BW 1.1 R t50 PA bs		
○	Dejan	Mijatović	EN 287-1	EN 287-1:2004 111 P BW 1.2 B t14 PF bs		
○	Radoslav	Petronijević	EN 287-1	EN 287-1:2004 111 P BW 1.3 B t14 PF bs		
○	Radoslav	Petronijević	EN 287-1	EN 287-1:2004 111 P BW 1.1 R t50 PA bs		

© 2008 Innovation center (Faculty of Mechanical Engineering, Belgrade University)
Kraljice Marije 16, Belgrade, Serbia | Phone: +381 (11) 330 23 46 | Fax: +381 (11) 337 03 64

Enter New

Name	
Company Type	○ ANB ○ ATB ○ Client
Is Active	[✓ No] ○ Yes
International ID	
International Prefix	
Accreditation Number	0
Country	Romania, Serbia, Bulgaria
State/Province	
City	
Postal Code	
Street Address	
Address Code	
Telephone	
Fax	
Email	

Search

First name		Last name	
Type	○ EN 287-1 ○ EN ISO 9606-2	Examination Company	Гоша Институт
Welding processes		Welding positions	
Material group(s)		Type of weld	
Material thickness (mm)		Outside pipe diameter (mm)	

○	Radoslav	Petronijević	EN 287-1	EN 287-1:2004 111 P BW 1.1 R t50 PA bs	Mon Aug 04 00:00:00 CEST 2008	Mon Aug 04 00:00:00 CEST 2008
○	Boban	Sretenović	EN 287-1	EN 287-1:2004 135 P BW 1.2 S t18 PF ss nb	Wed Sep 03 00:00:00 CEST 2008	Wed Sep 03 00:00:00 CEST 2008

Fig. 4. Visually hierarchical nature of the WP 2.0 interface

The next major change was adoption of more flexible interface methods. In the WP 1.0 system, an ad hoc inclusion of CSS and JavaScript was applied. In WP 2.0, a more systematic approach in the form of Ajax was applied, particularly the way Ajax is implemented in RichFaces in the form of the Ajax4jsf framework. The main objective was to speed up the data entry process by automating it. It is irritating for any user to have to enter the same data repeatedly. For example, in WP 1.0 the name of the city for every person living in the same town had to be entered repeatedly, because the interface had only HTML input fields or dropdown lists. An interface based on Ajax provides input fields with various data suggestion mechanisms based on asynchronous data retrieval methods that make data entry a less frustrating experience.

A very important feature of RichFaces is 'skinnability' – the ability to quickly change graphical appearance of the GUI. The look of the application is not a superficial system property. Users often have problems to shift their manual work process to a computer-based process if the computer-based process does not resemble the previous work environment. At this time it appears that the WP 2.0 users have accepted the new GUI, but we have already received a few suggestions that the printed-paper version of the certificate should precisely match the existing certificate form if we wish a company to start considering our system.

6. Database Reimplementation

The database reimplementation process was straightforward. The database structure in the WP 1.0 system was defined using the SQL language. So, for the new version the same code was used with minor modifications relating to data types (Table 1). During data import into the MySQL DBMS, a maximum row size error emerged as the problem. After reviewing the database structure it was revealed that the programmers were too comfortable in Oracle DBMS environment and had defined unnecessarily large fields. After reducing their size, a new database was implemented without any loss in functionality.

Table 1. Data type differences in the WP 1.0 and WP 2.0 systems

WP 1.0	WP 2.0
Oracle	MySQL
NUMBER	INT, FLOAT
NVARCHAR2	VARCHAR
ORDIMAGE, ORDDOC	LONGBLOB

Another point in using MySQL technology is Unicode support. The default system installation does not support Unicode. It is necessary to define Unicode support during the installation process and afterward to manually change a few `.ini` files. After that the Unicode support worked flawlessly.

7. Code Reimplementation

The actual code implementation exploited much of the code from the previous software version. Fortunately, the composition of the programming team was the same in both projects. During the technology selection process, the team recognized many similarities between the old and new technologies, particularly on the code level, and they used their findings to reuse as much code as possible. While in the previous project the programmers applied code that suited their needs and used many non-standard extensions, the restriction that only code conforming with standard definitions could be applied was imposed for the development of WP 2.0. The principal reason for this decision was to ensure the creation of the code that would be adaptable to future changes.

The programmers established an analogy between the systems based on the Model–View–Controller (MVC) pattern (Fig. 5). The Model part consisted of the Oracle ADF Business Component in the WP 1.0 system and of the EJB 3.0 JPA in the WP 2.0 system (Table 2). The first phase of the code reimplementation consisted of building the JPA entities. This process is automated in Eclipse IDE and requires the programmer to point to appropriate tables in the database for the creation of all relevant classes. Some modifications were necessary in the case of the `int` data type that needed modification to `integer` to support a `null` value. The `timestamp` data type needed manual conversion. Session bean functionality was written from scratch. Since the SQL code was located in the core of the software structure, the programmers established an analogy with the WP 1.0 code and managed to reuse it. The lack of the View object in the EJB 3.0 implementation required additional effort in programming entity objects that combined the data from different tables. Much of the Java code relating to business rules, as well as logical and numeric control was reused in the new system.

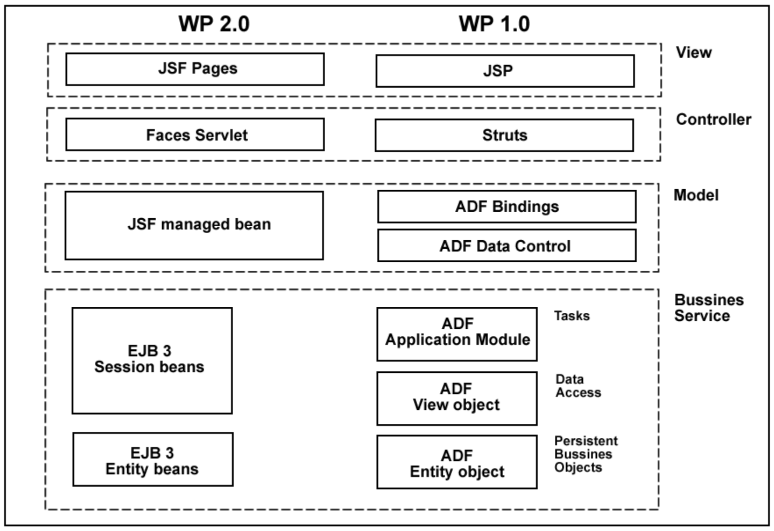


Fig. 5. Model–View–Controller (MVC) pattern of the WP 1.0 and WP 2.0 systems

On the view-controller side of the system, an analogy between Struts and JSF was established. This phase of the code reimplementation required a great deal of new programming.

8. Design Method

The redesign process shares many similarities with the design process. The main difference is that the design process starts from the need for a new system, while the redesign process is the consequence of dissatisfaction with the existing system. However, in respect to methodology, the same method can be applied in both cases.

The initial software design method depicted the process as the linear sequence of activities [15]. This is known as the “waterfall model”. Main phases (i.e., analysis, design, coding, testing, and operation) followed each other and continuation with the next phase was allowed only after the previous phase has been complete. Although this method guaranties an excellent final product, it was recognized that often it is impossible to define all of the requirements in advance because the users cannot fully describe their needs before they have had some preliminary experience with the software system.

The next generation of design methods has adopted iterative and incremental development as their foundation [16]. The main process phases have remained same. However, instead of progressing from phase to phase, during any particular iteration a usable part of the software is provided to users so that the designers can include any feedback in the next cycle. The learning occurs both during the development and during the use of the system. In this way a system evolves toward full implementation.

A further advance in software design methods is the Model-Driven Architecture (MDA) [17] concept. The basic notion of MDA is to separate the design process from the definition of the software architecture and implementation technology. User requirements are defined using a computation independent model (CIM). CIM describes the problem and the proposed solution without reference to the software technology that will be used. CIM serves as the foundation for the platform independent model (PIM) concept. PIM describes the software model but does not specify a concrete software platform. In the next phase the platform specific model (PSM) is defined, serving as a basis for the implementation model that guides the actual coding.

The most recent design methodology, referred to as agile software development [3], derives from the iterative model. It represents a family of methods that define a conceptual framework for software development. The methods are organized around a central set of basic principles such as communication, simplicity, feedback, courage, and respect. Communication and respect imply that communication, participation, respect, and trust must exist among stakeholders in the software development process. These principles play a crucial role during the user requirements definition phase. For the sake of simplicity the communication among participants must be effortless, comprehensible, and based on models that all team members understand. The feedback is based on the use of models and visual representations that enable prompt comments on particular ideas and their rapid adaptation to real needs. Courage involves the ability of the development team members to make prompt decisions. They need to have sufficient courage to completely change the development direction or abandon particular ideas if they lead to unacceptable results.

This paper describes the decisions and actions that were taken during the system redesign as a linear sequence. However, the actual process was far from linear. It was marked by many decision loops. Frequently, final assessments forced us to reconsider our initial assumptions. In one case, a daring decision was required to terminate an endless loop. As a method, the redesign process was more similar to a technique referred to as "cooperative gaming of invention and communication" [18] then any of the design methods described above.

We had three objectives as a goal: a) to deliver software, b) to create software amenable to change, and c) to create an adaptable redesign methodology. We spent two thirds of the allocated time to achieve the second and third goal, and as a result managed to achieve the first goal in one third of the allocated time. A crucial success factor was the improved understanding among project members that came as the result of the technology selection process. In addition, daring decisions were made each time a system architect recognized that the decision loop was entering a third cycle. In that case, a technology that appeared most promising was selected, but the decision was not forced onto the other team members. Instead, the entire decision cycle was described to all of the team members, and the reasons for the selection of the particular technology were presented. At that point, suggestions from other team members were accepted only if they proposed arguments that would result in the immediate acceptance of another technology (i.e., arguments yielding to a new decision cycle were rejected).

Communication played a major role in the redesign process. In the development of the WP 1.0 system, both programmers and welding specialists forced to express their viewpoints. As a consequence, the system grew too large and too complex. The system architect assumed a role in this project that architects traditionally have in building design – the role of the mediator between interacting parties. Having had training and experience as both an engineer and software developer, he understood that engineers and programmers often use the same words to represent different concepts. In that way, he largely increased the speed with which information moved between the welding specialists and the programmers. He also used his understanding of both disciplines to effectively emphasize by analogy to building design and construction that the WP 2.0 system should serve people instead of forcing users to become servants of the technology, even if that requires a large effort on the part of the engineers and programmers. In the communication process, no deadlocks were allowed: if one approach to a team member did not succeed, then another approach was devised until a shared understanding among the team members was achieved.

9. Conclusion

In retrospect we can say that we succeeded in producing usable software and preparing for the inevitable future software changes in the available period. The software is now in the deployment phase in two companies in Serbia, and under consideration for use in a few companies in Romania. The main problem that is still delaying widespread use is the need to maintain both the manual and the computer-based certification management processes during the deployment phase.

We have no evidence that our preparation for future software changes has been effective. Only case, so far, requiring change was the development of the system's demonstration version. Since it is quite difficult to explain to probable users how to install the SQL server, web server, and all of the Java libraries and code, we decided that the creation of a working USB demonstration version was an appropriate solution. Since the development was conducted in the Microsoft Windows environment, the live USB implementation required the WP 2.0 system to work under the Linux OS. The whole implementation took one day to complete, providing us with mild optimism that our system will be able to adapt to future changes.

10. Acknowledgments

The authors wish to express their appreciation to all members of the E!2774 project, who provided many results that were used during the design and development of the WP 2.0 system. Professor Stojan Sedmak was a constant driving force behind the development of both the WP 1.0 and WP 2.0 systems. Horia Dascau, Zlatan Milutinović, and Aleksandar Živković provided many insightful comments on the actual experience of using the system.

References

1. Zhang, J., Lin, Y., Gray, J.: Generic and Domain-Specific Model Refactoring using a Model Transformation Engine. In Beydeda, S., Book, M., Gruhn, V., (eds.): Model-driven Software Development - Research and Practice in Software Engineering, Springer, Berlin Heidelberg. 199-217 (2005)
2. Opdyke, W. F.: Refactoring: A Program Restructuring Aid in Designing Object-Oriented Application Frameworks. PhD Thesis, University of Illinois at Urbana-Champaign, USA. (1992)
3. Cockburn, A.,: Agile Software Development. Addison Wesley, Boston. (2000)
4. Radović, N., Radaković, Z., Đurović, A., Sedmak, S., Jandrlić, A., Golubović, D., Zrilić, M., Prokić-Cvetković, R., Popović, O., Milović, Lj., Rakin, M., Engh, E.: Welders passport-program structure and application. In Proceedings of the 1st South-East European Welding Congress - Welding and joining technologies for a sustainable development and environment, Timisoara, Romania, 260-263, (2006)

5. Sedmak S., Svetel I., Đurović A., Kovačević T.: Welders passport information system. Zavarivanje i zavarene konstrukcije, vol. 53, no. 3, 117-123, (2008) (in Serbian)
6. Quintino L., Ferraz R., Fernandes I.: The EWF Integrated Manufacturer Certification System. In Proceedings of the 1st South-East European Welding Congress - Welding and joining technologies for a sustainable development and environment, Timisoara, Romania, 207-213, (2006)
7. McConnell, S.: Code Complete, Second Edition. Microsoft Press, Redmond. (2004)
8. Highsmith, J.: Adaptive Software Development. Dorset House, New York. (2000)
9. http://www.visualsvn.com/server/
10. http://www.eclipse.org/subversive/
11. https://glassfish.dev.java.net/javaee5/persistence/
12. http://www.icefaces.org/main/home/
13. http://www.jboss.org/jbossrichfaces/
14. https://glassfish.dev.java.net/
15. Royce, W.W.: Managing the development of large software systems: concepts and techniques. Proceedings of the 9th international conference on Software Engineering, Monterey, 328- 338, (1987)
16. Larman, C., Basili V.R.: Iterative and Incremental Development: A Brief History. IEEE Computer, vol. 36, no. 6, 47-56, (2003)
17. Mellor, S.J., et al.: MDA Distilled: Principles of Model-Driven Architecture. Addison Wesley, Boston, (2004)
18. Cockburn, A.: The End of Software Engineering and the Start of Economic-Cooperative Gaming. Computer Science and Information Systems, vol. 1, no. 1, 1 -32, (2004)

Igor Svetel graduated at the Faculty of Architecture, University of Belgrade in 1986, and completed his Ph.D. at the same school in 2003. He is research associate at Innovation center, Faculty of mechanical engineering in Belgrade. His research interests are related to the design methodology applied both to the fields of software development and AEC, and application of ICT in architecture.

Aleksandar Đurović is the programmer specialized in the development of MVC web applications using J2EE, Oracle, and Open Source technologies. His interests include Oracle and MySQL databases and object-relational mapping in Oracle ADF, EJB3, and Hibernate using JSF, Spring, Struts, and Struts2 on Oracle AS, Glassfish, JBoss, and Tomcat application servers. He is also developing mods for internet forums developed in PHP.

Vencislav Grabulov graduated at the Faculty of Technology and Metallurgy, University of Belgrade in 1975, and completed his Ph.D. at the same school in 1995. He served as the President of the Serbian Welding Society and Chief executive of Authorized National Body for education of welding personal. He is now general manager of the Institute for Testing of Materials in Belgrade. His research interests are related to the: research and development of high strength steels and Al alloys, materials testing, fracture mechanics testing of metallic materials and composites, welding and fabrication by welding, behavior of materials during welding, welding metallurgy and weldability, and nondestructive testing methods.

Received: April 22, 2009; Accepted: January 19, 2010.