Modeling Requirements Elicitation Process for Web Applications

Marian Cristian Mihăescu[1], Cosmin Stoica Spahiu[1], Mihai Mocanu[1]
and Bogdan Logofatu[2]

[1]University of Craiova, Faculty of Automatics, Computers and Electronics
Software Engineering Department, Bvd. Decebal, Nr. 107, 200440, Craiova, Dolj, Romania
{mihaescu, stoica_cosmin, mocanu}@software.ucv.ro
[2]University of Bucharest, CREDIS Department
Bd. M. Kogalniceanu 36-46, Sector 5, Bucuresti, Romania
logofatu@credis.ro

Abstract. Requirements engineering plays a critical role within a software development process. Studies have revealed a lack of systematic processing of requirements due to high number of activities that need to be accomplished in this phase. There are many reasons for this situation. One of the most difficult tasks is to model requirements elicitation process. This is the first and one of the most important steps from the implications point of view in the nest steps of the development process and in the quality of the obtained software. This paper presents a requirements elicitation technique that has been successfully used in the development process of a web application. The direction of improvements are represented by requirements traceability and their modeling during elicitation phase.

Keywords. Requirements elicitation, traceability, modeling, web application.

1 Introduction

Ideally, software development, based on any of the well-established life cycle models described in [9] or [10], is linear, starts from scratch, and its phases can be (logically) delimited. No matter the software development model referred, distinct identifiable phases such as, requirements specification and analysis, architectural (overall) design, component design, implementation, component integration, validation and verification, code maintenance and evolution, can be extracted. If and how much are they really separated and easy to reveal, this is a difficult matter related to the specific and goals of the software development process applied.

In practice, software development must deal with the strong tendency to overlap development phases, due perhaps to social issues, and problem domain traditional "ways of doing things", little considered in the past within the discipline of software engineering. The facts are clear: software developers make mistakes, clients change their requirements while the software product is being developed, errors in operating software cannot be avoided a.s.o. The Winburg mini-case study [10] proved all these issues in the most convincing manner. From the implementation of the first version of

Mihaescu M., Stoica C., Mocanu M. and Logofatu B.
Modeling Requirements Elicitation Process for Web Applications.
DOI: 10.5220/0004464900640072
In *Proceedings of the 2nd International Workshop on Enterprise Systems and Technology (I-WEST 2008)*, pages 64-72
ISBN: 978-989-8111-50-0

a software product, continuing to the episode when a fault is found (i.e. due to a slow product operation), and a new design has to be adopted (i.e. using a faster algorithm), or an episode when the requirements change (i.e. because the accuracy has to be increased), the only thing that is "stable" and could be easily noted is a *recurrence of problems*.

Requirements engineering, based on requirements specification, elicitation and analysis, is not only the first, but also one of the most critical, knowledge-intensive activities of software development [1] The most basic question in requirements engineering is how to find out what users really need. Research has shown that in general many large projects fail because of inadequate requirements and specifically that poor execution of elicitation will almost guarantee that the final project is a complete failure. Since project failures are so rampant [2], it is quite likely that improving how the industry performs elicitation could have a dramatic effect on the success record of the industry [3]. Improving requirements elicitation requires us to understand it first. Although many papers have been written to define elicitation, or to prescribe a specific technique to perform during elicitation, nobody has defined yet a unified model of the elicitation process that emphasizes the role of knowledge.

The motivations of this paper can be seen in our effort to reach such a model. We'd like to make the necessary steps in establishing and describing (documenting) the requirements engineering phase, with an emphasis put on requirements elicitation, based on our experience in large software projects. Furthermore, we want to illustrate how we assessed the correctness of all these steps, based on a realistic software product development. In this paper, our specific aim is to discuss all the improvements regarding requirements elicitation that have been tested under real circumstances when developing an e-Learning platform called Tesys [4].

The organization of the rest of the paper is as follows. First, the paper presents an overview of the requirements engineering process (involving requirements specification, elicitation and analysis). Then it presents briefly the Tesys application platform. Next, it discusses the proposed improvements to requirements elicitation, regarding traceability and modeling of requirements in the elicitation process, and the benefits regarding verification and validation The paper concludes with a discussion of the proposed solutions and makes recommendations on how requirements elicitation could proceed.

2 Requirements Specification, Elicitation and Analysis

The purpose of the requirements engineering process, which involves requirements definition (specification), elicitation and analysis is to document the requirements for the next phases to be implemented during the software process according to the specific aims of the project. From a social point of view, this should be a collaborative process involving domain expertise from the client and software expertise from the contractor part. The key concept behind this step consists of the separation of application requirements from business/process requirements, which later on permits to link these to design objects. The up-front separation of application from business requirements really helps to clarify and focus on each aspect of the user's

requirements, or even helps to determine who should be asking for information about the requirements.

The requirements analysis methodology should cover the entire cycle, from the initial requirements-gathering phase through the separation phase where requirements and non-requirements are set apart. The overall steps required to define an approved set of requirements, and reach closure of the glossary definitions, should be:

1. Collection of 'raw' requirements – This step may start with the domain area experts (in this case the physicians) to write a few pages of text describing the problem to be solved. All relevant sources of possible requirements must be collected. Multiple sources of domain knowledge must be found, to allow the verification of sources. Specialized software tools, such as EasyRM Requirement Management Suite (http://www.easy-rm.com/) or AnalystPro (http://www.analysttool.com/) could be used from this point onwards.
2. Refining of requirements - The objective of this step is to identify the key business concepts, and their properties. Each requirement must be processed in turn for:

– Exclusion for all the following processing steps of any requirements, which are outside the scope of the mission statement,
– Decomposition into sub-requirements, containing only one concept.
– Naming convention – making sure that the name of the requirement reflects the requirement content.
– Redundancies elimination and resolving of duplicate requirements.
– Adjustment of 'priority' (initially all requirements will have the same low priority) and 'status'. Once all requirements have at least the status of 'Approved' then the requirement work has been completed to the point where the requirement set can be passed to the modelers of both teams (client team and implementation team) – possibly to act from here as a 'joint' team, for the creation of the conceptual models.

3. Establishing of reference sources for requirements – Many requirements are directly derived from mission papers, and other conceptual documents. A document manager tool can allow links to be established to these documents.
4. Creation of a document reference (the so-called 'requirements document'), in a bottom-up manner, with a short description giving an overview of what the other documents contain, and a location field used to point to the actual document – this can be filled with a path in the file system, or an URL. As the number of documents may increase, it is essential to classify these in a hierarchical manner, into folders required for specific subject areas.
5. Pass the 'requirements document' to system modelers, to check with the conceptual model they created in parallel with steps 3 and 4.

The end product of this succession of steps should be a requirements document and a correct system conceptual model which should allow the successful development of a software framework and possibly an open source architecture for the software product.

3 Tesys Application Platform

An e-Learning platform that represents a collaborative environment for students, professors, secretaries and administrators has been designed and developed. Secretary users manage sections, professors, disciplines and students. The secretaries have also the task to set up the structure of study years for all sections. The main task of a professor is to manage the assigned disciplines. The professor sets up chapters for each assigned discipline by specifying the name and the course document and manages test and exam questions for each chapter. The platform offers students the possibility to download course materials, take tests and exams and communicate with other involved parties like professors and secretaries.

The Tesys platform has initially been designed and implemented only with core functionalities that allowed involved people (learners, course managers, secretaries) to collaborate in good conditions. The requirements engineering followed an ad-hoc process that informally followed the classical life-cycle: elicitation, modeling, analysis, validation, verification and management. The involved parties were represented by three parties: development team, beneficiaries and end-users. Firstly, a prototype that implemented main functionalities has been developed. The requirements were elicited and negotiated between development team and beneficiary. After prototype has been deployed the e-Learning system has been populated with data and users. The beneficiary was the one that kept a close relation with end-users and closely looked the effectiveness of the platform.

The e-learning platform consists of a framework on which a web application may be developed. On server side we choose only open source software that may run on almost all platforms. To achieve this goal Java related technologies were employed.

The model is represented by DBMS (Data Base Management System) that in our case is represented by MySQL [11]. The controller, which represents the business logic of the platform is Java based, being build around Java Servlet Technology [12]. As Servlet container Apache Tomcat 5.0 [13] is used.

This architecture of the platform allows development of the e-learning applica-tion using MVC architecture. The view tier is template based, WebMacro [14] tech-nology being used. WebMacro is also a Java based technology the link between view and controller being done at context level. The separation between business logic and view has great advantages against having them together in the same tier. This decoupling makes development process more productive and safer. One of the biggest advantages of not having business logic and view together is the modularity that avoids problems in application testing and error checking.

In the figure 2 there are presented the main software components from the MVC point of view. MainServlet, Action, Manager, Bean, Helper and all Java classes represent the Controller. The Model is represented by the DBMS itself while the Webmacro templates represent the View. The model is built without any knowledge about views and controllers.

The business logic of the application uses Java classes. As it can be seen in figure 2, there are four levels of dependency between classes. The levels are: servlets, actions, managers and beans. Servlets level has so far two of them: MainServlet and DownloadServlet.

The MainServlet first job first job is to initialize application's parameters. For this purpose the init() method is used. Firstly, there is initialized a pool of database connections. Helper classes like ConnectionPool or ExecuteQuery based on the information from database.properties configuration file conduct this process. In the database configuration file there are set the address of MySQL server and the username and password of MySQL user that is used.

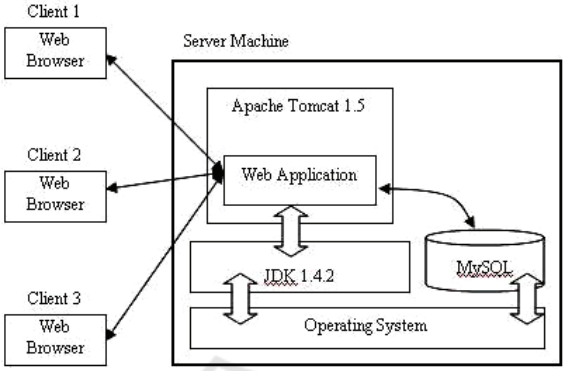

Fig. 1. Software architecture of the platform.

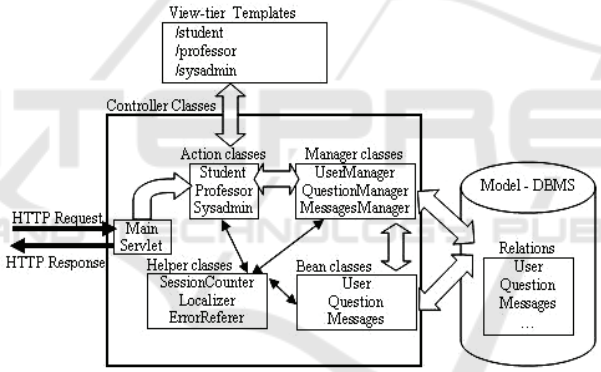

Fig. 2. Software components of the application from MVC point of view.

Another important part of software architecture regarding software development process is unit testing. For this purpose JUnit [15] is used. Unit tests are created for running the critical code like creating of a test, computing the result, saving the questions from the test, saving the test result, computing time for test. To accomplish this regressive testing is used. For each chain of actions a scenario is defined. If the computed result matches the expected result it means the test passed. Otherwise, it means something is wrong with the code because it does not behave like it supposed to. Whenever a method is added, test cases are written trying to have a full coverage of the code. There are created batch files that build the code experimentally and continuously and run all the tests. Similarly, a scheduled job runs the nightly build of all the code from the staging area and runs all tests.

The platform is currently in use on Windows 2003 Server machine. This platform has three sections and at each section four disciplines. Twelve professors are defined and more than 650 students. At all disciplines there are edited almost 2500 questions. In the first month of usage almost 500 tests were taken. In the near future, the expected number of students may be close to 1000.

4 Improvements in Requirements Elicitation

We present improvements regarding two issues: traceability and modeling of requirements in the elicitation process.

A requirement is defined as an object with his own status and life cycle. The status is determined by the set of values of fields. We define the following set of fields:

Id – uniquely identifies the requirement;
Role – defines the role to which the requirement addresses;
Activity – defines the activity to which the requirement addresses;
Status – there were defined three states: INWORK, SOLVED and VERIFIED;
Solver – person responsible for implementing the requirement;
Memo – text that represents a short summary about the requirement.
Date – represents the date when the action has been executed on the requirement
This structure ensures the traceability of the requirement.

- The improvements in requirements elicitation is analyzed from the following points of view [6]:Time and Effort allocation: how requirements elicitation is distributed over time
- Artifacts produced by requirements elicitation: various deliverables like data, effects, results, documents, etc, resulted from requirements elicitation process usage.
- Requirements elicitation Activities: what activities produce artifacts including the requirements specification.
- Disciplines and automation: specify major areas of concern that can influence
- Requirements elicitation, technology and management tools
- Roles: Various roles in RE and differences between them.

The process of modeling the requirements is described in figure1.

The requirements phase has its own life cycle. The specialty literature proves that is difficult to give a general description of requirements activities. In the 80's Krasner identified five phases: need identification and problem analysis; requirements determination; requirements specification; requirements fulfillment; and requirements change management. Another approach presented by Jarke and Pohl in 1994 propose a three-phase cycle of elicitation, expression and validation.

It seems that different approaches use different labels for the requirements activities and this brings about one of the critical problems in requirements engineering: lack of a systematic process. The main problem is that requirements Engineers involves people that communicate with other people. The communication is hard due to the lack of a common scientific language and knowledge. The

misunderstandings from this kind of communication are translated directly in wrong application development.

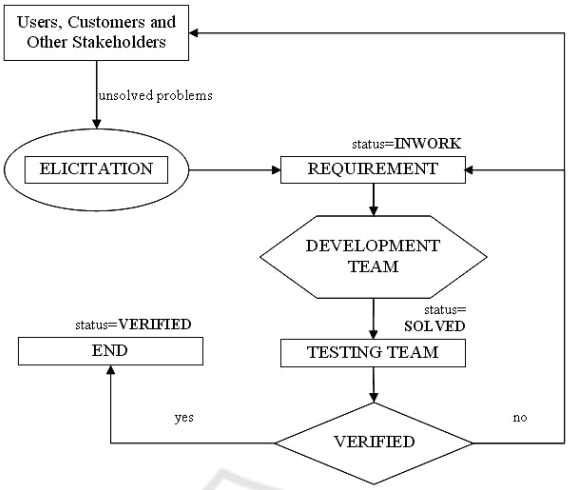

Fig. 3. The process of modeling requirements.

The existing problems for requirements elicitation have been grouped into three categories as follows [8]:

- problems of scope (the system edge capabilities are not well defined, unnecessary design information may be given);
- problems of understanding (users don't know exactly what they need and don't know the compter limitations, the users frequently skip "obvious" information, there can be conflicts views from different users
- requirements are often vague and untestable (problems of volatility, requirements evolve over time)

Although this area was not researched enough, there are a series of techniques developed to solve these problems. Traditional methods include brainstorming, interviewing and use of questionnaires[8]. The latests methods for requirements engineering include techniques for information gathering, modeling and representation of information.

Requirements engineering relies fundamentally on verification and validation as a way of achieving quality by getting rid of errors, and as a way of identifying requirements.

One benefit from structuring requirements is the use of automation for verification of requirements. The requirements may be inspected such that verification is performed by using well established checklists. The checklists are applied to the requirements by a well established process.

Modeling requirements in a custom structured form provides the opportunity for analyzing them. Analysis techniques that have been investigated in requirements engineering include requirements animation, automated reasoning, consistency, and a variety of techniques for validation and verification that are further discussed.

Validation is the process of establishing that the requirements and derived structures provide an common and accurate base for involved persons (developers and beneficiaries). Explicitly describing the requirements is a necessary precondition not only for validating requirements, but also for resolving conflicts between developers and beneficiary.

Difficulty of requirements validation comes from many sources. One reason is the problem itself is philosophical in nature. This makes the formalizing process hard to define. On the other hand, there is a big difficulty in reaching agreement among involved persons dew to their conflicting goals. The solution to this problem is requirements negotiation. These will attempt to resolve conflicts between involved parties without necessarily weakening satisfaction of each person's goals.

Structuring requirements brings a big advantage for validation and verification in case of changing requirements. As all successful systems, our e-Learning platform evolves. This means that when a functionality changes because of beneficiary and developer negotiated such a change, this transition needs to be done with minimum of effort. For this, requirements have to be traceable and this feature is accomplished by proposed structuring.

5 Conclusions

In this paper, there were presented the main challenges in requirements engineering and especially requirements elicitation. There were presented solutions regarding traceability and modeling of requirements during elicitation phase.

Proposed solutions were tested during Tesys e-Learning platform software development process. It has been also presented the initial requirements engineering process that was used when the prototype has been developed as compared to the improved one.

Proposed solutions come to support a big effort of software globalization development process since the application is rapidly growing in size. More than this, the business logic complexity, degree of heterogeneity among assets is increasing.

Other benefit is that there may be created pools of requirements based on functionality at role level and even with a higher granularity at activity level. This will have a big impact on future decisions regarding what parts of software to be out-sourced in the effort of globalization.

From requirements point of view there were presented three improvements. Finally, there presented the benefits brought by our structuring to verification and validation processes. The proposed structure ensures traceability of requirements, such that as the system evolves the requirements are still properly managed.

References

1. Gottesdeiner, E.: Requirements by Collaboration, Addison-Wesley, (2002)
2. Standish Group, The Chaos Report, www.standishgroup.com, (1995)
3. Hofmann, H., and F. Lehner: Requirements Engineering as a Success Factor in Software Projects, IEEE Software, 18, 4 (2001)

4. Burdescu, D.D., Mihăescu, M.C.: Tesys: e-Learning Application Built on a Web Platform, Proceedings of International Joint Conference on e-Business and Tele-communications, Setubal, Portugal (2006)
5. Ann M. Hickey, Alan M. Davis, "Requirements Elicitation and Elicitation Technique Selection: A Model for Two Knowledge-Intensive Software Development Processes," *hicss*, p. 96a, 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 3, 2003
6. Bhavani Palyagar, Frank Moisiadis, "Validating Requirements Engineering Process Improvements - A Case Study," *rev*, p. 9, First International Workshop on Requirements Engineering Visualization (REV'06 - RE'06 Workshop), 2006
7. Daniela E. Herlea Damian, "Challenges in Requirements Engineering", Requirements E, Springer, Springer, 2003, vol. 8, no.3, pp. 149-160
8. Michael G. Christel , Kyo C. Kang, "Issues in Requirements Elicitation", Technical Report, 1992
9. Sommerville I., *Software Engineering*, 7th Ed., Pearson –Addison Wesley, 2004
10. Schach S.R., *Object-Oriented and Classical Software Engineering*, 6th Ed., McGraw Hill, 2006
11. Randy Jay Yarger, George Reese, Tim King, "Managing & Using MySQL, Second Edition", O'Reilly, 2002.
12. Jason Hunter, "Java Servlet Programming, 2nd Edition", O'Reilly, 2001.
13. Chanoch Wiggers, "Professional Apache Tomcat", Wiley Publishing, 2003.
14. Faulk, S. "Software Requirements: A Tutorial, Software Engineering", Los Alamitos, CA: IEEE Computer Society Press, 1996.
15. A. Sutcliffe, S. Fickas, and M. M. Sohlberg. PC-RE a method for personal and context requirements engineering with some experience. Req. Eng. J., 11(3):1–17, 2006.