# Effective requirement specification as a precondition for successful software development project

Željka Požgaj
Faculty of Economics, University of Zagreb
J.F. Kennedy Square 6, 10000 Zagreb, Croatia
Tel: +385 1 238 32 77, Fax: +385 1 233 56 33
E-mail: zpozgaj@efzg.hr

Hrvoje Sertić
Research & Development Center, Ericsson Nikola Tesla
Krapinska 45, 10000 Zagreb, Croatia
Tel: +385 98 416 777, Fax: +385 1 365 36 10
E-mail: hrvoje.sertic@etk.ericsson.se

Marija Boban
Faculty of Law, University of Split
Domovinskog rata 8, 21000 Split, Croatia
Tel: +385 98 955 08 05, Fax: +385 21 393 597
E-mail: marija.boban@law.pravst.hr

**Abstract.** *Software development is activity closely connected with requirements. To enable development of software systems that satisfies most of customer demands in this work we propose methodologies for obtaining efficient requirement management infrastructure on software development project. We propose methodologies for requirement discovery and organization according to competence areas available on the project and involved risks. We also propose methodology for managing change of requirements during software development project in order to enable prosperous project conclusion even if major requirement change occurs. Presented methodologies provide means of effective requirement management that can significantly improve quality of complex software systems.*

**Keywords.** Software development, requirement management

## 1. Introduction

Continuous technological progress has enabled development of complex software solutions with new, previously unrealizable features and possibilities. Development of such systems is connected with extremely wide area of requirements that such software solution must fulfill. Enormous quantity of requirements that must be satisfied in order to build quality software solution is the source of many development problems. There is a huge number of development projects that have failed simply because they were unable to define and track software requirements [3]. This work is focused to present effective means of requirement specification and methodologies that can be used in order to support and improve requirement management techniques on software development projects. Therefore, with this work we propose methodologies for dealing with software requirements that should be used in order to develop successful software solution. Methodologies proposed in this work are intended to establish requirement management environment and connect requirements with developer's competences and project risks in order to enhance realization of complex software solutions. Since the conditions on today's global software market are rapidly changing, change of requirements during software development is common situation. Consequently, we have also proposed methodology for dealing with requirement change that enables prosperous project conclusion even if major requirement change occurs. Presented methodologies are dedicated to provide effective requirement management practice and to support development of complex software solutions according to the customer demands.

## 2. Software requirements

Software requirements can be defined as formal descriptions of customer demands on software system solution [3]. These demands are usually captured as text statements about capabilities of software system. Software requirements can be divided into two main categories [3], [4]: (1) functional and (2) non-functional. Functional requirements capture and describe system functionality, while non-functional requirements capture all other demands on software solution such as reliability, performance, usability and quality of service. Functional requirements are usually considered as more important because they describe how software system should work. Non-functional requirements are often forgotten because system development is focused only to system functionality. Non-functional requirements are important because software system functionality (defined by functional requirements) must be performed on acceptable level that is defined by non-functional requirements or the system is considered as unsuccessful. Software system functionality is not satisfying if it is performed well (in functional terms), but too slow or unreliable. Therefore, it cannot be stated that one kind of requirements is more important than other because software system should provide defined functionality on highest possible usability and performance level.

### 2.1. Software requirement descriptions

Since software requirements are textual descriptions of customer demands they can be created on different ways. The main problem connected with software requirement descriptions is the content of description, not the form. The discovery of software system's requirements is long and complicated process that must be considered extremely important in order to develop successful software solution [1]. However, formal definition of template for software requirement description is important precondition for efficient requirement management because software requirements must be described in form that is easy to understand and use [4]. Therefore, with this work we propose definition of template that should unify all software requirements descriptions for software development project. Beside purely textual definitions, system's functional requirement should be illustrated with various models. Use-case modeling technique [5], [7] is one of commonly used techniques for creating functional requirement models. Use-cases are textual descriptions of interaction between systems and its users focused to capture events that are interchanged during system execution [5]. Requirement modeling techniques such as use-case modeling are important because they provide explanation of system requirements in form suitable for software development. But it must be noted that requirement description is the base for all requirement-modeling techniques that are only a tool used to lighten requirement understanding. Textual descriptions of software requirements contain information about actual customer demands that should be used to guide software development in order to build right software solution.

### 2.2. Software requirements and software development

Software requirements should be used in all phases of software development in order to provide guidelines for development activities [3] [4].Today is common to use iterative approach to software development. In iterative approach, life cycle of software development is divided into couple of small project called iterations [1]. Each iteration contains all phases and activities of software development, from requirement definition and modeling, to implementation and testing. The result of each iteration is build upon results of previous iterations; so that software solution is build in incremental manner [1]. Software requirements must be used to guide such approach to software development, because each iteration is based on selection of software requirements that will be implemented [1]. This approach to software development demands useful definitions of software requirements because software requirements are used to plan software development [8]. However, there are many problems connected with effective use of software requirements for guiding software development such as bad requirement definition, wrong requirement selection and change of requirements during software development [1], [2]. Beside these problems, software requirements are often badly organized and difficult to understand. To solve these problems, we propose methodologies for efficient requirement management.

## 3. Methodologies for efficient requirement management

Since software requirements are used in all phases of development project there are many possible problems caused by poor requirement definition, improper requirement use and requirement change during software development project. Methodologies presented in this work focus to complete requirement life cycle, from discovery and description to requirement use in all phases of development project. These methodologies describe activities that should be performed in order to achieve efficient requirement management environment and to effectively use requirements for guiding complete software development.

### 3.1. Methodology for requirement discovery and description

The first proposed methodology focus to the process of requirement discovery and definition. The process of requirement discovery is highly important because all set of requirements must be discovered in order to build software solution that satisfies customer demands [6]. Therefore, to avoid problems connected with late requirement discovery such as system architecture change (system architecture can be defined as software system organization [1] or structure of significant components interacting through interfaces [8]) and development of wrong software system, a specific approach to requirement discovery proposed with this methodology should be taken. Fig. 1 presents activities defined with methodology for requirement discovery using standard object oriented notation's (UML – *Unified Modeling Language*) activity diagrams [9]. Methodology starts with activities intended for assessing system scope in order to examine target system problem area and to define global project scope in terms of used technology. After that, a competence based project infrastructure should be established. This means that developer's competences and knowledge should be assessed because each member of software development team should be responsible for tasks that are closely connected with his competences. After that, system's requirement scope should be divided into areas according to

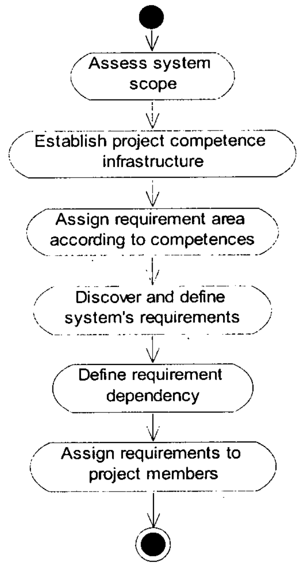
Assess system scope
Establish project competence infrastructure
Assign requirement area according to competences
Discover and define system's requirements
Define requirement dependency
Assign requirements to project members

**Figure 1: Methodology for requirement discovery**

competences available on the project and assigned to project members. With this approach, each project member is responsible for requirement discovery and definition according to his competence area, i.e. each project member should discover requirements only in assigned requirement area. By dividing requirement area according to people's competences we have created requirement discovery infrastructure that allows each project member to focus on requirements only in requirement areas closely connected with his knowledge and competences. After requirement discovery, whole set of requirements should be analyzed in order to find dependency among them. Information about requirement dependency is important for selection of requirements to be realized, because it is common that realization of one requirement demands realization of one or more other requirements [1], [2]. By capturing dependency information, project team can easily choose requirements for realization. When the whole set of requirements is identified, described and analyzed in terms of dependency, requirements should be assigned to particular project members according to their competences and become their responsibility. Therefore, presented methodology divides requirements according to knowledge areas on software development project to improve requirement discovery and definition process because project team members will better understand and capture requirements if they are closely connected with their competences.

## 3.2. Methodology for requirement organization

The second proposed methodology is focused to requirements use during software development life cycle. Methodology presented on Fig. 2 shows activities that should be performed to create requirement infrastructure suitable for all phases of development project. Methodology starts with requirement analysis and definition of requirement impact to system architecture. Project members should analyze only the requirements that are part of their responsibility. Requirement analysis should be focused to requirement properties and requirement impact to system's architecture based on requirement dependencies defined by methodology for requirement discovery. Requirement analysis should be followed by definition of requirement probability of change in order to select requirements with high probability of change. After in-depth requirement analysis project risks connected with each requirement realization should be identified. Usually there is high number of risk impacts on software development projects connected with the use of advanced technologies [6]. Therefore, risks on software development project should be listed and in-depth described according to defined requirement areas. This approach will improve risk identification procedure because risks will be explored according to connected requirements. We propose risk identification for each requirement regardless to probability of risk materialization and amount of risk impact, because probability of risk materialization usually changes during software development [2]. Beside risk properties, risks should be defined according to requirement impact and probability of change. When all requirement properties are identified, they should be described together with requirement itself. Proper requirement definition and description of all requirement properties are precondition for requirement organization [2]. With this methodology we propose to sort requirements in each requirement category (defined according to the competence areas available on software development project) upon risks connected with them. With this approach, requirements with highest risk impact will be first selected for realization, therefore enabling the most important risks to be solved first, when the costs connected with risk materialization are still small [8]. Proper requirement definition and organization proposed with this methodology will enable easy requirement understanding and significantly improve requirement use on software development project. The methodology for requirement organization doesn't stop with requirement selection for realization and implementation. Since the requirements are assigned to particular project members, they are responsible for continuous requirement supervision and tracking of requirement realization. With this approach, requirements should be continuously supervised during complete development [1], [2], [6].

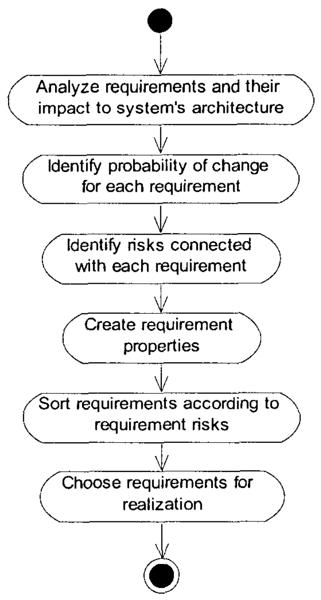
Analyze requirements and their impact to system's architecture
Identify probability of change for each requirement
Identify risks connected with each requirement
Create requirement properties
Sort requirements according to requirement risks
Choose requirements for realization

**Figure 2: Methodology for requirement organization**

## 3.3. Methodology for requirement change management

Requirement change during software development is common to most software development projects [1] and serious threat to success of software development project because requirement change can have significant impact to complete system architecture [2], [7]. Therefore, requirement change management is essential activity on any software development project. Requirement change management is considered as important development activity, but still many development project fail because of requirement change [2]. In order to successfully complete software development project in case of requirement change, methodology presented on Fig. 3 should be continuously applied during all development phases. Requirements with high probability of change (identified with methodology for requirement organization) should be continuously tracked and supervised in order to discover requirement change as soon as possible. This is closely connected with requirements assigned to project members, because each project member is responsible for supervision of assigned requirements. Therefore, each project member should continuously track requirements assigned to him and promptly identify possible requirement change. When the indications for requirement change occur, requirements should be closely analyzed and requirement change identified. After requirement change identification, all consequences of requirement change must be dissected in order to define facts for decision about requirement change implementation. Requirement change analysis should be focused to two main areas: (1) requirement change impact to other requirements [6] and to (2) definition of requirement change actions. These actions should be defined for all development areas that are under requirement change impact to enable immediately response to requirement change. Definition of requirement change actions prior to decision about requirement change realization is important because project management team should have complete information about requirement change impact and consequences to software system. The purpose of requirement change analysis actions is to provide sufficient information for decision about requirement change realization [1]. Therefore, project management team should decide for each requirement change request according to data provided by requirement change analysis to implement or to refuse requirement change. With this methodology, we propose to define special iteration on software development project that will implement change of requirements in all phases of software development, from re-work of requirement specification to implementation and testing. We strongly suggest to assess complete project status after any serious requirement change implementation in order to supervise requirement implementation impact to the complete software system. Presented methodology improves requirement analysis and focus to requirement change impact to complete system architecture in order to improve development capabilities in terms of requirement change realization [1].

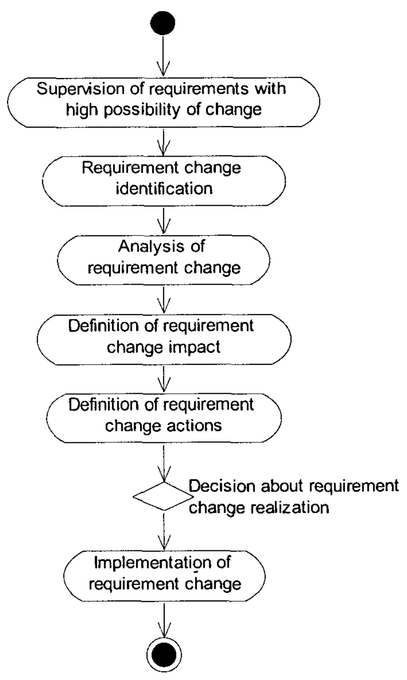
Supervision of requirements with high possibility of change
Requirement change identification
Analysis of requirement change
Definition of requirement change impact
Definition of requirement change actions
Decision about requirement change realization
Implementation of requirement change

**Figure 3: Methodology for requirement change management**

## 3.4. Relationship and dependency between proposed methodologies

Proposed methodologies are closely connected by means of requirements and development life cycle because results of activities defined with methodology for requirement discovery and description are input to activities defined with methodology for requirement organization. Methodology for requirement change management depends on results from first two methodologies, properly defined requirements and corrects these requirements in case of change. Therefore, proposed methodologies form a complete approach to requirements use on software development project.

## 4. Evaluation of proposed methodologies

In this work we have proposed methodologies for requirement management based on our experience from several software development projects. In order to verify proposed methodologies we have applied them on real software development project, development of M2M (*Machine to Machine*) communication system based on GPRS (*General Packet Radio Service*) technology in Ericsson Nikola Tesla Company. We have started application of proposed methodologies when project was in initial state. Project had six developers involved beside project manager. First of all we evaluated competences of these developers and defined four main competence categories: (1) *GPRS technology* (2) *Operating systems* (3) *Hardware platforms* (4) *Software development languages and tools*. Those were also main requirement discovery areas assigned to project members. Because of initial project phase and small number of competence categories, we have discovered that there are many dependencies between requirements from different categories, for example, there were several requirements from *GPRS technology* category that depended on *Hardware platform* requirements. Other activities in requirement discovery and requirement organization methodology were performed according to definition and produced efficiently organized software requirements for our system. Since our project was cancelled after pre-study phase because our organization didn't find business case for project realization we haven't approached any requirement change demand. Methodology for requirement discovery and description together with methodology for requirement organization resulted with most of system's requirement discovered and defined. Those requirements were used to create system prototype that was functional before project was cancelled. Based on achieved results we can confirm that proposed methodologies enable efficient requirement discovery and use on software development project with one limitation: requirement categories must be correctly defined on project start. If requirement categories are wrongly defined, many problems may occur because members of project development team will not be able to efficiently use their competences. Therefore, proposed approach is efficient only with correct competence and requirement area definition.

## 4. Conclusion

Requirements are the heart of software development. In order to be successful, software solution must satisfy many different kinds of requirements demanded by customer. Requirements should be used in all phases of software development, as a guideline that will lead to software solution capable of satisfying all customers needs. There are many problems connected with requirement discovery, definition and use on software development project that impact complete software development and result with software system that fails to meet customer demands. Requirement change is connected with another set of problems during software development because requirement change usually requires change of software architecture. To solve these problems and improve requirement use on software development project we propose three methodologies presented in this work. These methodologies form best practice for requirement management according to our experience and should be performed in all phases of software development project. Proposed methodologies improve all aspects of requirement use on software development project and ameliorate quality of software solutions by providing infrastructure for software development according to customer demands.

## 5. References

[1] Larman C. Applying UML and Patterns. New York: Prentice Hall PTR; 2002.

[2] Robertson S, Robertson J. Mastering the Requirements Process. New York: Addison-Wesley; 1999

[3] Leffingwell D, Widrig D, Yourdon E. Managing Software Requirements: A Unified Approach. New York: Addison-Wesley; 1999.

[4] Wiegers E. K. Software Requirements. San Francisco: Microsoft Press; 2002

[5] Bittner K., Spence I. Use Case Modeling. New York: Addison- Wesley; 2002.

[6] Gause D, Weiberg G. Exploring Requirements: Quality Before Design. London: Dorset House; 1989.

[7] Cockburn A. Writing Effective Use Cases. New York: Addison- Wesley; 2001.

[8] Rational Unified Process, online documentation, http://www.rational.com [10/02/2003]

[9] UML 1.4 Specification http://www.omg.org [10/02/2003]