Volume 3, No. 3, March 2012

Journal of Global Research in Computer Science

ISSN-2229-371X

TECHNICAL NOTE

Available Online at www.jgrcs.info

A TECHNICAL DISCUSSION ON REQUIREMENT ENGINEERING WITH AN EXAMPLE

Onkar Nath Pandey *1, D.B.Ojha2

* Research Scholar, CMJ University

pandeyonkarnath@yahoo.co.in 1

2Department of Science& Technology, Mewar University, Rajasthan, INDIA

ojhabrat@gmail.com2

Abstract: Requirements engineering is concerned with the early phases of software development. This paper presents historical background and need of Requirements Engineering. More and more problems with software projects were being traced back to problems in the requirements process, therefore, the importance of requirements engineering was realized. Various definitions are discussed. Subsequently, importance of Requirement Engineering is explained with cases.

INTRODUCTION

Creating software is not an easy task. Since the conferences in 1967 and 1968 by the NATO Science Affairs Committee on software problems[1], the proper engineering of software has become a important topic.

Requirements engineering is concerned with the early phases of software development. More and more problems with software projects were being traced back to problems in the requirements process, therefore, the importance of requirements engineering was realized. International Symposium on Requirements Engineering was held in1993 at San Diego, California, USA. Thus, the requirements engineering became increasingly accepted.

Various techniques, methods and frameworks have been developed to address issues in this area. However, the application of these research results in industry has not been very extensive. It was reported in 1994 that majority of the failures causing unsuccessful software projects were related to poor requirements engineering practices. Software requirements are still poorly analyzed, documented [2].

As the first phase of software development, requirements engineering needs to handle problems that are complex, uncertain, constantly changing, and under strict environmental or organizational constraints. While the life cycles of software projects become longer and the projects' size increases, requirements engineers face more challenges to analyze and manage the software requirements. Support for requirements engineering practices are necessary.

In spite of the demand for requirements engineering tools, currently available tools are limited to requirements management and modeling [3]. In order to improve the requirements engineering practices in software industry, it is necessary to develop requirements engineering tools that offer more advanced support from the early stage.

Software plays an important role in today's systems, therefore, more cost and effort areinvested in software development. People have realized since the 1970s that getting the right requirements is a prerequisite for successful software development [4]. The challenges in software development shift from implementation to defining the behavior of software in its environments. Requirements engineering has emerged in order to better understand how the software interacts with the other parts of the system.

Although a lot of effort has been put in requirements engineering, industry organizations are still practicing poor software requirements engineering. Transferring this to industry organizations has encountered significant problems. It has been recognized that the lack of intelligent support for requirements engineering is one of the main reasons to block applying requirements engineering methodologies.

DEFINITIONS

The IEEE software engineering glossary [5] defines a requirement as:

a. A condition or capability needed by a user to solve a problem or achieve an objective;
b. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents
c. A documented representation of a condition or capability as in (1) or (2)

Requirements should only state what a system is supposed to do rather than to describe the way to do it. The requirements can be classified into two categories: functional requirements and non-functional requirements [6,7]. The definitions of these two terms is given [8].

A *functional requirement* is "a software requirement that specifies a function that a system or its component must be capable of performing. These are software requirements that define behavior of the system, that is, the fundamental process or transformation that software and hardware components of the system perform on inputs to produce outputs"

A *non-functional requirement* is "a software requirement that describes not what the software will do, but how the software will do it. Examples include software performance requirements, software external interface requirements, software design constraints and software quality attributes.

Non-functional requirements are usually difficult to test; therefore, they are usually evaluated subjectively."

As functional requirements are focusing on the software product itself, the non-functional requirements can also include process and external requirements [4]. There are several definitions for the term ***requirements engineering***. The different definitions reflect the different aspects of requirements engineering. Some artifacts, which were not addressed in earlier definitions, are explicitly included in later ones. In 1990, requirements engineering was defined below as [5].

***Requirements engineering* is :**

a. The process of studying user needs to arrive at a definition of system, hardware, or software requirements;
b. The process of studying and refining system, hardware or software requirements".

This definition highlights three important elements in requirements engineering. They are "user needs", "studying", and "definition", which can be interpreted as input, task, and output of the process. This definition omits the importance of other stakeholders besides the user. Requirements engineers need to cooperate with all the stakeholders to develop a more complete list of requirements.

In [9], the following definition is given *Requirements engineering* is "a systematic process of developing requirements through an iterative, co-operative process of analysing the problem, documenting the resulting observations in a variety of representation formats and checking the accuracy of the understanding gained".

This definition focuses more on the activities in requirements engineering including observations, analyzing, checking the accuracy, and documenting. This definition realized that there is more than one role involved in requirements engineering. It also recognizes that certain activities occur again and again throughout the process. "Iterative" and "co-operative" became two key features of requirements engineering. In 1997, requirements engineering was defined as follows [7]

Requirements engineering is "a term that has been invented to cover all of the activities involved in discovering, documenting, and maintaining a set of requirements for a computer-based system".

This is a very broad definition that includes all the activities related to the discovery and maintenance of the requirements. In this definition, requirements engineering is not restricted to the beginning of software development. It could extend to the whole software life cycle in order to maintain the requirements of the software. Another definition of this term is included in [10]

Requirements engineering is "the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behaviour, and to their evolution over time and across software families".

This definition shows the relationship between software engineering and requirements engineering. It implies that reuse of requirements can be applied to software families. According to this definition, the process of requirements engineering would continue even after the life cycle of a software product in order to facilitate the requirements of succeeding products.

The different definitions of requirements engineering show that requirements engineering has to be expanded in all aspects including the scope, the life cycle, the activities in the process and the roles involved in the process.

Finally, a definition of *requirements engineering* **process** is given in [11]. The requirements engineering process is "a structured set of activities which are followed to derive, validate and maintain a systems requirements document". This definition highlights that the requirements engineering process is composed of a set of structured activities.

RESULT AND DISCUSSION

Importance of requirements engineering:

Requirements increase the understanding of the proposed system. Faults in the requirements can impact all the successive phases of software development. Two real industry projects are discussed in this section to show how insufficient or incorrect requirements engineering can impact the whole software project.

In a project of the London Ambulance Service Computer-Aided Dispatch (LAS CAD) [12], a computer-based system was developed to coordinate the ambulances according to emergency calls. When the full Computer Aided Dispatch system was expanded to serve all of London for the first time, users of the system were faced with a number of changes to the way in which they did their work. Neither control staff nor ambulance crews could understand the system completely.

The end result was ineffective service such that the system was completely abandoned.

In the beginning of this project, the software development team had not cooperated with the end users because of the political and organizational issues. There was no communication between the end users and other stakeholders to make agreements on how the system would be installed and how it would affect their work. The system did work as specified, but it failed because these specifications did not include all the stakeholders.

Another case is an example of incorrect requirements management causing trouble in software development [13]. The project was carried out by Software Assurance Technology Center (SATC). In the beginning of the project, the project team focused on decomposing high-level requirements to low-level requirements and put the links between different requirements into a database. As the projects proceeded, the requirements increased. At the point, when there were 1,500 level 3 requirements, 6,000 level 4 requirements, and 19,000 links, the requirements became a big mess.

As a result, requirements engineers had to manually transfer important information from the old tool This case shows that requirements management in software Development is a challenging task. Another lesson that can be learned from this case is that tools are very important for requirements engineering.

CONCLUSION

The primary measure of success of a software is the degree to which it meets the purpose for which it was intended. The software requirements engineering is the process of discovering that purpose.

REFERENCES

[1] P. Naur, and B. Randell, Software Engineering: A Report on a Conference Sponsored by the NATO Science Committee, NATO, 1969

[2] H. Kaindl, S. Brinkkemper, and J.A. Bubenko Jr, "Requirements engineering and technology transfer: obstacles incentives and improvement agenda," RequirementsEngineering, vol. 7, no. 3, pp. 113-123, 2002.

[3] Q. Zhang, and A. Eberlein, "Architectural design of an intelligent requirements engineering tool," 2003 Canadian Conference on Electrical and Computer Engineering, CCECE'03, Montreal, QC, 2003.

[4] A. Eberlein, Requirements Acquisition and Specification for Telecommunication Services PhD Thesis, Swansea, UK: University of Wales, 1998.

[5] IEEE, 610.12-1990 IEEE Standard Glossary of Software Engineering Terminology (ANSI), [1-55937-067-X] [SH13748-NYF], New York: IEEE Press, 1990P35 .

[6] L. Chung, Representing and Using Non-Functional Requirements: A Process-Oriented Approach, PhD Thesis, ON: University of Toronto, 1993.

[7] I. Sommerville, and P. Sawyer, Requirements Engineering – A Good Practice Guide, New York: Wiley, 1997 .

[8] R.H. Thayer, and M. Dorfman, System and Software Requirements Engineering,Washington: IEEE Computer Society Press, 1990 .

[9] P. Loucopoulos, and V. Karakostas, System Requirements Engineering, New York: McGraw-Hill, 1995.

[10] P. Zave, "Classification of research efforts in requirements engineering," ACM Computing Surveys, vol 29, no. 4, pp. 315-321, 1997.

[11] G. Kotonya and I. Sommerville, Requirements Engineering: Processes and Techniques, New York: Wiley, 1997.

[12] L. Macaulay, Requirements Engineering, London: Springer, 1996.

[13] T. Hammer, and L., "Automated Requirements Management – Beware HOW You Use Tools, An Experience Report," International Conference on Requirements Engineering, (ICRE'98), Colorado, CO, 1998.

Short biodata of all the author

Dr.D.B.Ojha: Ph.D from Department of Applied Mathematics, Institute of Technology, Banaras Hindu University, Varanasi (U.P.), INDIA .The degree field is Optimization Techniques In Mathematical Programming. The major field of study is Functional Analysis. He is working at Mewar Universit, Rajasthan as Professor. He has the authored/coauthored of more than 200 publications in technical journals and conferences.

Mr. O. N. Pandey did his B.Tech.(IIT, Kanpur)1968, ME(IIT,Roorkee)1970. His specialization in Instrumentation, Control and Industrial Automation under UNIDO Fellowship from USA, UK, ITALY and JAPAN and also membership of UNIDO fellowship, Numerical Control Society.