Volume 3, No. 9, September 2012
Journal of Global Research in Computer Science
ISSN-2229-371X
RESEARCH PAPER
Available Online at www.jgrcs.info

REQUIREMENT ENGINEERING: AN APPROACH TO QUALITY SOFTWARE DEVELOPMENT

Dhirendra Pandey[1], Vandana Pandey[2]
*[1]Department of Information Technology Babasaheb Bhimrao Ambedkar University, Lucknow
Prof.dhiren@gmail.com
[2]Department of Computer Science Dr. C. V. RamanUniversity, Bilaspur
Vandanadubey7@gmail.com

Abstract: The requirement engineering is the process of collection of requirements and further, implements it to the software development process. It is important for every organization to develop quality software products that satisfy the user's needs. To achieve this goal we have to apply requirement engineering practices in every step of software development process. We use requirement engineering practices because requirement engineering is most important phase of software development process and with the help of requirement engineering practices we collect user's requirement and implement them in software development process. The purpose of this paper is to give an idea to how requirement engineering is necessary for software development and how requirement engineering influences the software development process. In this paper we analyze the requirement engineering process for designing quality software products and also describe the importance of requirement engineering.

INTRODUCTION

The term requirement engineering are used to describe a systematic process of developing requirements through an iterative co-operative process of analyzing problem, documenting the resulting observation in a variety of representation formats, and checking the accuracy of the understanding gained. Requirements engineering is a transformation of business concerns into the information system requirements. Therefore, we can define requirements engineering as [1]: Systematic approach to eliciting, organizing, and documenting the requirements of the system, and a process that establishes and maintains agreement between the customer and the project team on the changing requirements of the system. The requirements engineering is the first phase of software engineering process, in which user requirements are collected, understood, and specified. Requirements engineering is recognized as a critical task, since many software failures originate from inconsistent, incomplete or simply incorrect requirements specifications. Many of the most common, most serious problems associated with software development are related to requirement.

CLASSIFICATION OF REQUIREMENTS ENGINEERING PROCESS

In this section we present the overall framework within which requirements engineering takes place. The result of the requirements engineering phase is documented in the requirements specification. The requirements specification reflects the mutual understanding of the problem to be solved between the analyst and the client. The requirements specification servers as a starting point for the next phase, the design phase. To achieve well-defined documents containing the user requirements that satisfy these prerequisites, we can distinguish three processes in requirements engineering [2]. These processes involve iteration and feedback (figure 1). Several classifications have been proposed for requirements engineering. [3]:

a. Requirement elicitation
b. Requirement specification
c. Requirement verification and validation.

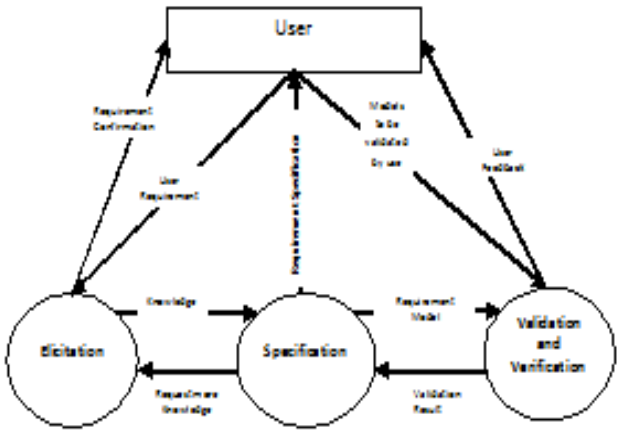

Figure1: Requirement Engineering Process

Requirements Elicitation:

Requirements elicitation is about understanding the problem. In general, the requirements analyst is not an expert in the domain being modeled. Through interaction with domain specialists, he has to build himself a sufficiently rich model of that domain. The fact that different disciplines are involved in this process complicates matters. It many cases, the analyst is not a mere outside observer of the domain modeled, simply eliciting facts from domain specialists.

Requirements Specification:

Once the problem is understood, it has to be described in the requirements specification document. This document describes the product to be delivered, not the process of how it is developed.

Requirements Validation and Verification:

Once the problem is described, the different parties involved have to agree upon its nature. We have to ascertain that the correct requirements are stated(validation) and that these requirements are stated correctly(verification).

REQUIREMENTS ENGINEERING AND SOFTWARE DEVELOPMENT LIFE CYCLE

Many models exist for the system and/or software life cycle, the series of steps that a system goes through from first realization of need through construction, operation, and retirement [4]. Almost all models include one or more phases with a name like "requirements analysis" or "user needs development". Many models require generation of a document called, or serving the function of, a requirements specification. Even those that do not call for such a document, for example Jackson System Development, have a product such as a diagram of diagrams that incorporate or express the user's needs and the development objectives [5]. Waterfall models are briefly discussed subsequently and the way requirements engineering fits into them are presented. Among the most extensively used models are baseline management and the waterfall, on which baseline management is based, [6].

In this models, as shown in Figure 2 and 3, determination of requirements should be complete, or nearly so, before any implementation begins. Baseline management provides a high degree of management visibility and control has been found suitable for developments of very large size in which less complex methods often fail, and is required under many software development systems. This model, however, has been somewhat discredited, because when large complex systems are developed I practice it is usually impossible to develop an accurate set of requirements that will remain stable throughout the months or years of development that follow completion of the requirements.

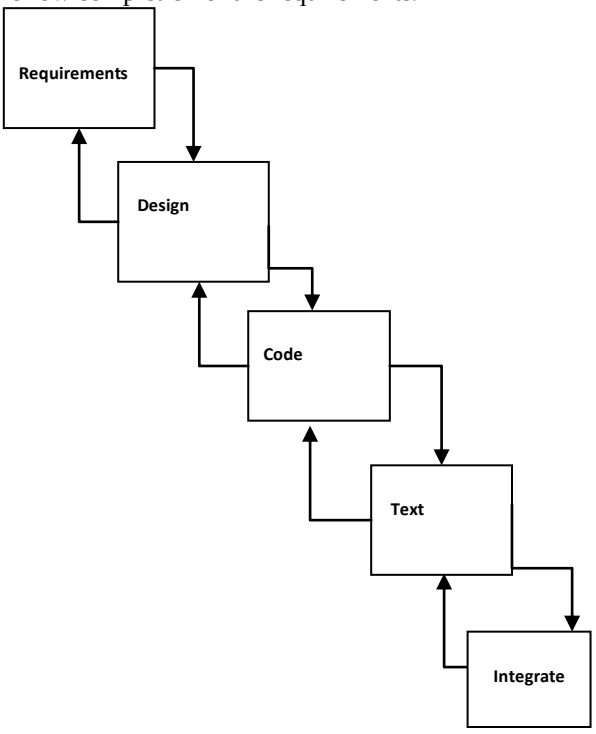

Figure 2: The baseline management and waterfall models

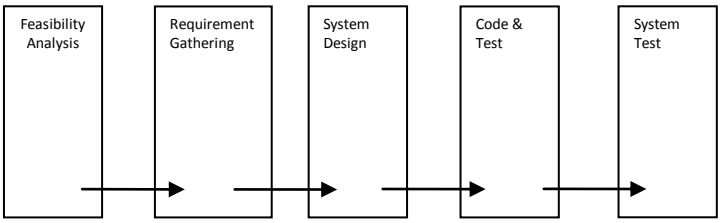

Figure 3: The Linear Software Development Lifecycle

REQUIREMENT ENGINEERING PRACTICES

The principles of requirements engineering described above are valid and important, but for practical application additional specifies are needed. These specifies ate provided by methods and tools. A method, sometimes referred to as a methodology, describes a general approach; a tool, usually but not always automated, provides a detailed, step- by –step approach to carrying out a method.

Methods:

Requirements analysis methods may be roughly divided into four categories, as shown in Figure 4. The categorizations should not be regarded as absolute: most methods have some of the characteristics of all the categories, but usually one viewpoint is primary.

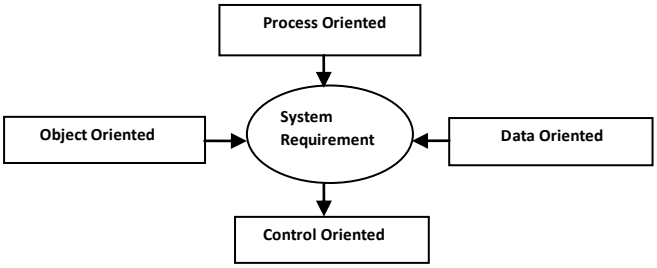

Figure 4: Categories of requirements analysis methods

Process-oriented methods take the primary viewpoint of the way the system transforms inputs into outputs, with less emphasis on the data itself and control aspects. Classical structured analysis and fits into his category, as do structured analysis and design technique [7,8], and formal methods such as Vienna development method and Z (A Formal Specification Method). Data – oriented methods emphasize the system state as a data structure [9, 10]. While structured analysis and structured analysis and design techniques have secondary aspects of the data viewpoint, entity-relationship modeling and Jackson system development are primarily data oriented. Control-oriented methods emphasize synchronization, deadlock, exclusion, concurrence, and process activation and deactivation. Structured analysis and design techniques and the real-time extensions to structured analysis [11 , 12] are secondarily control oriented. Flowcharting is primarily process oriented. Finally, object-oriented methods base requirements analysis on classes of objects of the system and their interactions with each other.

The number of tools that support requirements engineering is growing rapidly, and even the most cursory survey is beyond the scope of this paper. Nevertheless, some discussion of the characteristics of requirements engineering

tools, and trends in these characteristics, is in order. One of the researchers has classified requirements tools as follows:

a. Graphical editing
b. Traceability
c. Behavior modeling
d. Databases and word processing
e. Hybrid

IMPORTANCE OF REQUIREMENTS ENGINEERING PRACTICES

The primary measure of success of a software system is the degree to which it meets the purpose for which it was intended. Broadly speaking, software systems requirements engineering is the process of discovering that purpose, by identifying stakeholders and their needs, and documenting these in a form that is amenable to analysis, communication, and subsequent implementation. There are a number of inherent difficulties in this process. Stakeholders (including paying customers, users and developers) may be numerous and distributed. Their goals may vary and conflict, depending on their perspectives of the environment in which they work and the tasks they wish to accomplish. Their goals may not be explicit or may be difficult to articulate, and, inevitably, satisfaction of these goals may be constrained by a variety of factors outside their control.

CONCLUSION

The most common, most serious problems associated with software development are related to requirement analysis. Begin from the term definition, we discussed the requirements engineering and its dimension. And finally, we analyzed the requirement engineering practices and give the importance of requirement engineering practices for designing quality software products from several viewpoints.

BIBLIOGRAPHY

[1]. Romi satria wahono,"Analyzing Requirement engineering Problem", IECI Janpan workshop, Japan, page 55-58, 2009

[2]. P. Loucopoulos and V. Karakostas: Software Requirements Engineering, McGraw-Hill,1995.

[3]. Davis, Alan M. ,private communication, 1996.

[4]. IEEE Standard 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology, IEEE, NY, 1990.

[5]. Cameron, John R. , "An Overview of JSD," IEEE Transactions on Software Engineering, Vol. 12, No. 2, pp. 222-240, 2011

[6]. Royce, Winston W., "Managing the Development of Large Software Systems," Proceedings, IEEE Wescon, August 1970. Reprinted in Proceedings, 9th International Conference on Software Engineering (Monterey, CA, pp 328-338, 2008.

[7]. Svoboda, Cyril P., "Tutorial on Structured Analysis," in System and Software Requirements Engineering, R.H. Thayer and M. Dorfman, eds., IEEE Computer Society Press, Los Alamitos, CA,1990.

[8]. Ross, Douglass T., "Structured Analysis (SA): A Language for Communication Ideas," IEEE Transaction on Software Engineering, Vol. 3, No. 1, pp. 17-29, January 2007

[9]. Bjoerner, Dines, "On the Use of Formal Methods in Software Development," Proceedings, 9th International Conference on Software Engineering (Monterey, CA, March 30-April 2, 1987), IEEE Computer Society Press, Washington, pp.17-29, 2006.

[10]. Norris, M., "Z (A Formal Specification Method). A Debrief Report," STARTS, National Computing Centre, Ltd., 1986. Reprinted in System and Software Requirements Engineering, R.H. Thayer and M. Dorfman, eds., IEEE Computer Society Press , Los Alamitos, CA, 1990.

[11]. Ward, Paul T., and Stephen J Mellor, Structured Development Techniques for Real-Time Systems (3 vols.). Prentice-Hall, Englewood Cliffs, NJ, 1985.

[12]. Hatley, Derek J., "The Use of Structured Methods in the Development of Large Software-Based Avionics Systems," AIAA Paper 84-2592, Sixth Digital Avionics Systems Conference (Baltimore,MD. December 3-6), 1984.

