Proceedings of the IECI Japan Workshop 2003
Chofu Bunka Kaikan Tazukuri, Japan

ISSN 1344-7491
pp. 55 - 58

ANALYZING REQUIREMENTS ENGINEERING PROBLEMS

Romi Satria Wahono
Department of Information and Computer Sciences, Saitama University
Lembaga Ilmu Pengetahuan Indonesia (LIPI)- Pusat Dokumentasi Informasi Ilmiah (PDII)
E-mail: romi@romisatriawahono.net
URL: http://romisatriawahono.net

Abstract. 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. In this paper we analyze the root problems of the requirements engineering from several viewpoints.

Keyword: *software engineering, requirements engineering*

1. Introduction

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. The Standish Group study noted that three most commonly cited factors that caused projects to be challenged [Leffingwell-00]:

- Lack of user input: 13 percent of all projects
- Incomplete requirements and specifications: 12 percent of projects
- Changing requirements and specifications: 12 percent of all projects

However, a correct, consistent and complete way to collect, understand, specify and verify user requirements is important and necessary.

We can summarize the issues discussed in the requirements engineering as the "rock" problem. The requirements that describes "bring me a rock", will be actually "bring me a small blue rock", or "bring me a spherical small blue rock." All the people can become frustrated by the problems of specifying an acceptable "rock". We have got to get it right the first time yet also provide for iterative process in which the customer ultimately discovers what kind of rock he wants.

In this paper we try to analyze the root problems of the requirements engineering from several viewpoints.

2. Concepts and Term Definitions

2.1. Requirement

The definitions of a requirement according to [IEEE-610.12] [IEEE-830] [IEEE-729] are:

1. A condition or capability needed by a user to solve a problem or achieve an objective.
2. 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.
3. A documented representation of a condition or capability as in 1 or 2.

The term user alluded to in this definition may be an end user of the system or a person behind the screen. However, it may also denote several classes of indirect users, such as people who do not themselves turn the knobs but rather use the information that the system delivers. It may also denote the customer (client) who pays the bill. During requirements engineering, different types of user may be the source of different types of requirement. The term user is used to denote both direct (end) users and other stakeholders involved in the requirements engineering process [Vliet-00].

The definitions of terms used in the requirements engineering that corresponds to the persons are as follows [IEEE-610.12].

Customer: The person, or persons who pay for the product and usually (but not necessarily) decide the requirements. In the context of this recommended practice the customer and the supplier may be members of the same organization.

Supplier: The person, or persons who produce a product for a customer. In the context of this recommended practice, the customer and the supplier may be members of the same organization.

__User__: The person, or persons who operate or interact directly with the product. The user(s) and the customer(s) are often not the same person(s).

2.2. Requirements Engineering

The term requirements engineering is 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:

A 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.

3. The Three Dimensions of Requirements Engineering

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 serves as a starting point for the next phase, the design phase. To achieve well-defined document containing the user requirements that satisfies these prerequisites, we can distinguish three processes in requirements engineering [Loucopoulos-95]. These processes involve iteration and feedback (Figure 1).

3.1. 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. In many cases, the analyst is not a mere outside observer of the domain modeled, simply eliciting facts from domain specialists.

3.2. 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.

3.3. 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).

4. The Problems of Requirements Elicitation

Problems of requirements elicitation can be grouped and classified into three categories [Christel-91]. These are problems of scope, problems of understanding, and problems of volatility. Leffingwell [Leffingwell-00] used another terms for this three scopes by the problems on “analyzing the problem” and “understanding the user needs”.

4.1. The Categories

- Problems of scope
  - ➢ The requirements may address too little or too much information.
  - ➢ The boundary of the system is ill-defined
  - ➢ Unnecessary design information may be given
- Problems of understanding
  - ➢ Problems of understanding within groups as well as between groups such as users and developers.

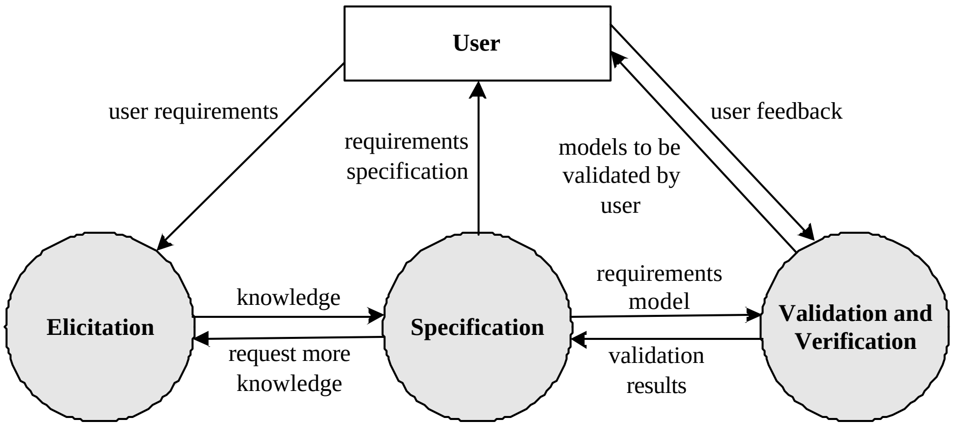

Figure 1: Requirements Engineering Process

  - Users have incomplete understanding of their needs
  - Users have poor understanding of computer capabilities and limitations
  - Analysts have poor knowledge of problem domain
  - User and analyst speak different languages
  - Ease of omitting "obvious" information
  - Conflicting views of different users
  - Requirements are often vague and untestable, e.g., "user friendly" and "robust"
- Problems of volatility
  - The changing nature of requirements.
  - Requirements evolve over time

Requirements elicitation is complicated by three endemic syndromes [Leffingwell-00].

1. The "yes but" syndrome stems from human nature and the users' inability to experience the software as they might a physical device.
2. Searching for requirements is like searching for "undiscovered ruin"; the more you find, the more you know remain.
3. The "user and the developer" syndrome reflect the profound differences between the two, making communication difficult.

4.2. The Techniques

These facts and problems give the researchers a place for discussing and proposing the requirements elicitation techniques. Some techniques are shown in the following:

- Interviewing and questionnaires
- Requirements workshop
- Brainstorming and idea reduction
- Storyboards
- Use cases
- Role playing
- Prototyping

5. The Problems of Requirements Specification

The previous Section was focused on the process of analyzing the problem, eliciting user needs, and collecting, documenting, and managing the desired product features. We have now arrived at the center of the requirements engineering dimension, the "specification process". A complete set of requirements can be determined by defining the system inputs, outputs, functions, attributes, and attributes of the system environment.

One of the most difficult challenges we face in the requirements specification process is making the requirements detailed enough to be well understood without overconstraining the system and predefining a whole host of things that may be better off left to others downstream in the process. The goal is to find the "sweet spot" or the balance point wherein the investment in requirement provides "just the right amount" of specificity and leaves just the "right amount of ambiguity" for others to resolve further downstream (see Figure 2).

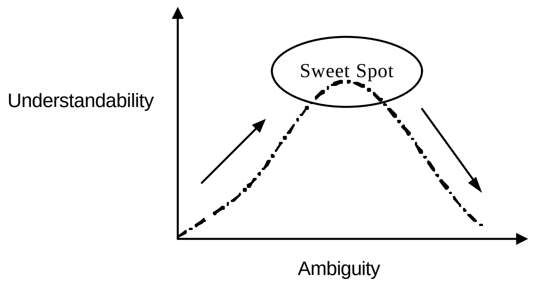

Figure 2: Ambiguity versus Specificity

Some specification techniques are proposed to solve this ambiguity problem. Use Case [Booch-99] [Rumbaugh-99] has achieved a degree of popularity and common use for expressing requirements for a system. Well implemented to the system that using the UML and object-oriented methods.

However, the most popular technique for documenting requirements was to use natural language and to simply write them all down in an organized fashion. This technique was revised and improved over the course of many projects, and eventually a number of standards developed for these documents, including IEEE (Institute of Electrical and Electronics Engineers) 830: Standard for Software Requirements Specification [IEEE-830].

IEEE Std 830-1998 [IEEE-830] also describes the characteristics of a good software requirements specification (eight quality measures). A software requirements specification should be:

1. Correct
2. Unambiguous
3. Complete
4. Consistent
5. Ranked for importance and/or stability
6. Verifiable
7. Modifiable
8. Traceable

If the description of the requirement is too complex for a natural language and if you can not afford to have the specification misunderstood, you should consider writing that portion of the requirements with a "technical methods" approach. Some technical specification methods are as follows:

- Pseudocode
- Finite state machines
- Decision trees
- Activity diagrams (flowcharts)
- Entity relationship models
- Object-oriented analysis
- Structured analysis

6. The Problems of Requirements Validation and Verification

Building the right system right depends on continually confirming that the development is on track and that the results are correct, as well as being able to deal with change during development.

Verification is the process of ensuring that development activities continually conform to the customer's needs. IEEE [IEEE-1012] defines verification as:

The process of evaluating a system or component to determine whether the products of a given phase satisfy the conditions imposed at the start of that phase.

Verification is supported by the use of traceability techniques to relate parts of our project to one another. By using traceability, we can verify that:

- All project elements are accounted for, and
- All project elements have a purpose

Validation demonstrates that the product conforms to its requirements and gains customer acceptance of the final result. IEEE [IEEE-1012] defines validation as:

The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements.

We use the validation techniques to ensure that:

- All project elements are properly tested
- All tests have a useful purpose

7. Conclusions

Many of the most common, most serious problems associated with software development are related to requirement. Begin from the term definition, we discussed the requirements engineering and its dimension. And finally, we analyzed the root problems of the requirements engineering from several viewpoints.

Acknowledgements

The author would like to thank to the Ministry of Education, Culture, Sports, Science and Technology of Japan for its financial and scholarship support.

REFERENCES

[Booch-99] Grady Booch, James Rumbaugh, and Ivar Jacobson, "The Unified Modeling Language User Guide", Addison-Wesley, 1999.

[Christel-91] Michael G. Christel and Kyo C. Kang, Issues in Requirements Elicitation, Technical Report CMU/SEI-92-TR-12, ESC-TR-92-012, September 1992.

[IEEE-729] Institute of Electrical and Electronics Engineers. IEEE Standard Glossary of Software Engineering Terminology. ANSI/IEEE Standard 729-1983, Institute of Electrical and Electronics Engineers, New York, 1983.

[IEEE-610.12] Institute of Electrical and Electronics Engineers, IEEE Standard Glossary of Software Engineering Technology, IEEE Std 610.12-1990, Institute of Electrical and Electronics Engineers, New York, 1990.

[IEEE-830] Institute of Electrical and Electronics Engineers, IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998, Institute of Electrical and Electronics Engineers, New York, 1998.

[IEEE-1012] Institute of Electrical and Electronics Engineers, IEEE Standard for Software Verification and Validation, IEEE std 1012-1998, Institute of Electrical and Electronics Engineers, New York, 1998.

[Leffingwell-00] Dean Leffingwell and Don Widrig: Managing Software Requirements – A Unified Approach, Addison Wesley, 2000.

[Loucopoulos-95] P. Loucopoulos and V. Karakostas: Software Requirements Engineering, McGraw-Hill, 1995.

[Rumbaugh-99] James Rumbaugh, Ivar Jacobson, and Grady Booch, "The Unified Modeling Language Reference Manual", Addison-Wesley, 1999.

[Vliet-00] Hans Van Vliet: Software Engineering - Principles and Practice, John Wiley & Sons, 2000.

Biography of Author

Romi Satria Wahono, Received B.E. and M.E degrees in Information and Computer Sciences in 1999 and 2001, respectively, from Saitama University, Japan. He is currently a Ph.D. candidate at the Department of Information and Computer Sciences, Saitama University, Japan and also a researcher at the Indonesian Institute of Science (LIPI). His current research interests include Software (Requirements) Engineering, Web Engineering, and Object-Orientation. He is a member of the ACM, IEEE Computer Society, The Institute of Electronics, Information and Communication Engineers (IEICE), Information Processing Society of Japan (IPSJ), Japanese Society for Artificial Intelligence (JSAI), and Indonesian Society on Electrical, Electronics, Communication and Information (IECI).