International Journal of Research
Available at https://edupediapublications.org/journals

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue-01
January 2018

Optimization of Requirements Engineering Analysis for Substitute Sets Of Functional Necessities

T.Prem chander & Dr.BV Ramana Murthy
[1]Research Scholar Rayalaseema University, Kurnool, AP, India
[2]Principal Stanley Womens Engineering College, Hyderabad, India
tudiprem@gmail.com ; drbvrm@gmail.com

Introduction

Purpose

The Requirements Specification Document (RSD) covers exactly the same ground as the requirements definition, but from the perspective of the developers. Where the requirements definition is written in terms of the customer's vocabulary, referring to objects, states, events, and activities in the customer's world, the requirements specification is written in terms of the system's interface. We accomplish this by rewriting the requirements so
that they refer only to those real-world objects (states, events, actions) that are sensed or actuated by the proposed system.

Document Conventions

In documenting the system's interface, we describe all inputs and outputs in detail, including the sources of inputs, the destinations of outputs, the value ranges and data formats of input and output data, protocols governing the order in which certain inputs and outputs must be exchanged, window formats and organization, and any timing constraints. Note that the user interface is rarely the only system interface; the system may interact with other software components (e.g., a database), special-purpose hardware, the Internet, and so on.

Next, we restate the required functionality in terms of the interfaces' inputs and outputs. We may use a functional notation or data-flow diagrams to map inputs to outputs, or use logic to document functions' pre-conditions and post-conditions. We may use state machines or event traces to illustrate exact sequences of operations or exact orderings of inputs and outputs. We may use an entity-relationship diagram to collect related activities and operations into classes. In the end, the specification should be complete, meaning that it should specify an output for any feasible sequence of inputs. Thus, we include validity checks on inputs and system responses to exceptional situations, such as violated pre-conditions.

Finally, we devise fit criteria for each of the customer's quality requirements, so that we can conclusively demonstrate whether our system meets these quality requirements.The criteria for validating the requirements are the characteristics that we listed below

- Correct
- Consistent
- Unambiguous
- Complete and relevant

Intended Audience and Reading Suggestions

We have been using the terms "verify" and "validate" throughout this chapter without formally defining them. In requirements validation, we check that our requirements definition accurately reflects the customer's – actually, all of the stakeholders' – needs. Validation is tricky because there are only a few documents that we can use as the basis for arguing that the requirements definitions are correct. In verification, we check that one document or artifact conforms to another. Thus, we verify that our code conforms to our design, and that our design conforms to our requirements specification; at the requirements level, we verify that our requirements specification conforms to the requirements definition. To summarize, verification ensures that we *build the system right,* whereas validation ensures that we *build theright system!*

Validation can be as simple as reading the document and reporting errors. We can ask the validation team to sign off on the document, thereby declaring that they have reviewed the document and that they approve it. By signing off, the stakeholders accept partial responsibility for errors that are subsequently found in the document. Alternatively, we can hold a walkthrough, in which one of the document's authors presents the requirements to the rest of the stakeholders, and asks for feedback. Walkthroughs work best when there are a large number of varied stakeholders, and it is unrealistic to ask them all to examine the document in detail. At the other extreme, validation can be as structured as a formal inspection, in which reviewers take on specific roles (e.g., presenter, moderator) and follow prescribed rules (e.g., rules on how to examine the requirements, when to meet, when to take breaks, whether to schedule a follow-up inspection).

Product Scope

Requirements validation determines whether the requirements are substantial to design the system. The problems encountered during requirements validation are listed below.

1. Unclear stated requirements.
2. Conflicting requirements are not detected during requirements analysis.
3. Errors in the requirements elicitation and analysis.
4. Lack of conformance to quality standards.

A number of other requirements validation techniques are used either individually or in conjunction with other techniques to check the entire system or parts of the system. The selection of the validation technique depends on the appropriateness and the size of the system to be developed. Some of these techniques are listed below.

1. **Test case generation:** The requirements specified in the SRS document should be testable. The test in the validation process can reveal problems in the requirement. In some cases test becomes difficult to design, which implies that the requirement is difficult to implement and requires improvement.

2. **Automated consistency analysis:** If the requirements are expressed in the form of structured or formal notations, then CASE tools can be used to check the consistency of the system. A requirements database is created using a CASE tool that checks the entire requirements in the database using rules of method or notation. The report of all inconsistencies is identified and managed.

3. **Prototyping:** Prototyping is normally used for validating and eliciting new requirements of the system. This helps to interpret assumptions and provide an appropriate feedback about the requirements to the user. For example, if users have approved a prototype, which consists of graphical user interface, then the user interface can be considered validated.

References

1. A Case Study on a Specification Approach Using Activity Diagrams in Requirements Documents http://ieeexplore.ieee.org/document/8048911/-- 26 September 2017- 2332-6441

[1] Roy GG, Woodings TL (2000) A framework for risk analysis in software engineering. In: Proceedings of the sevent listing and event priorityh Asia- Pacific software engineering conference (APSEC '00), IEEE

Computer Society Press, Washington, DC, USA, p 441
[2] Boehm BW (1991) Software risk management: principles and practices. IEEE Software, pp.32–41. doi:10.1109/52.62930
[3] B. W. Boehm. Software Risk Management: Principles and Practices. IEEE Software, pp.32–41, 1991.
[4] Risk Analysis as part of the Requirements Engineering Process YudistiraAsnar Paolo Giorgini March 2007
[5] Focusing on the Importance and the Role of RequirementEngineering Mina Attarha and Nasser ModiriAtta.mina@yahoo.com ;
[6] H.F. Hofmann and F. Lehner, "Requirements engineering as a success factor in software projects", *IEEE Software,vol 18, no* 4,pp. 58-66, 2001.

7] B.A. Nuseibeh and S.M. Easterbrook, "Requirements engineering: A roadmap", Proc. of the 22nd Intl. Conf. on Software Enginnering (ICSE '00), IEEE Computer Society Press, June 2000, pp. 35 – 46.
[8] T. Hall, S. Beecham and A. Rainer, "Requirements problems in twelve software companies: An empirical analysis", IEEE *Software, vol 149, no.* 5, pp. 153-160, 2002.
[9] M. Niazi and S. Shastry, "Role of requirements engineering in software development process: An empirical study", Proc. of the 7th Intl. Multi Topic Conf. (INMIC2003