Chapter 4

Requirements Management

CONTENTS

This page intentionally left blank.

Chapter 4

Requirements Management

4.1 Introduction

The goal of any endeavor is defined by its requirements. Requirements define the purpose of a project and are the foundation of the project plan. The stated requirements will determine what is to be produced or achieved. One of the primary reason software projects fail is because requirements are incomplete or incorrect. [1] If the requirements are not done right, the project will probably fail, no matter what is done in subsequent phases. [2] Incomplete or defective requirements result in inaccurate product descriptions and erroneous cost and schedule estimates, which in turn deliver an unusable product. What is not asked for will not be produced, and what is asked for will be produced even if it's not needed. Table 4-1 shows the relative costs of fixing software requirements defects in different phases of the project. Whatever it may cost to do things right in the requirements phase, it will be 3 to 1000 times more costly to fix later if not done right.

Phase in Which Fixed	Relative Cost
Requirements	1
Design	3 – 6
Coding	10
Development Testing	15 – 40
Acceptance Testing	30 – 70
Operations	40 – 1000

Table 4-1 Relative Costs of Fixing Requirement Errors [3]

If requirements are complex and not fully known up front, a life cycle that allows for the refinement of requirements, such as spiral development, should be used. More on this can be found in Chapter 2, Software Life Cycle.

Requirements engineering is a sub-discipline of systems engineering. This chapter presents an overview of requirements and requirements engineering. Because this is such a crucial aspect of any project, it is recommended that additional resources (section 4.6) be used for a greater insight and understanding.

4.2 Process Description

4.2.1 Requirements Definitions

Requirements define capabilities or conditions possessed by a system or a component needed to solve a problem or achieve an objective. They can be divided into two categories, functional and nonfunctional. Functional requirements describe what a system should do in response to a specific stimulus (e.g. when the power switch is first turned to "On" the radio will perform the standard self-check routines.) [4] This is shown in Figure 4-1.

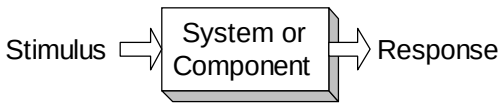

Figure 4-1 Functional Requirements

Nonfunctional requirements include performance and system constraints that affect development and design. They include categories such as these:

- Safety
- Performance
- Survivability
- Reliability
- Environment
- Implementation
- Support
- Security
- People

Requirements must be carefully derived through analysis of user needs and documented. A major point to remember is that requirements should specify *what* is to be done, not *how* it is to be done.

4.2.2 Requirements Process Overview

The requirements engineering process is the primary work of the Requirements phase of the project, but it continues throughout the project to some degree because of requirements changes in later phases. It consists of two major types of activities. The first is requirements development, which is the process of generating requirements from user inputs and analysis. The second is requirements management, which is all other requirements activities which are not part of the development process. Figure 4-2 shows the breakout of the major activities of requirements engineering.

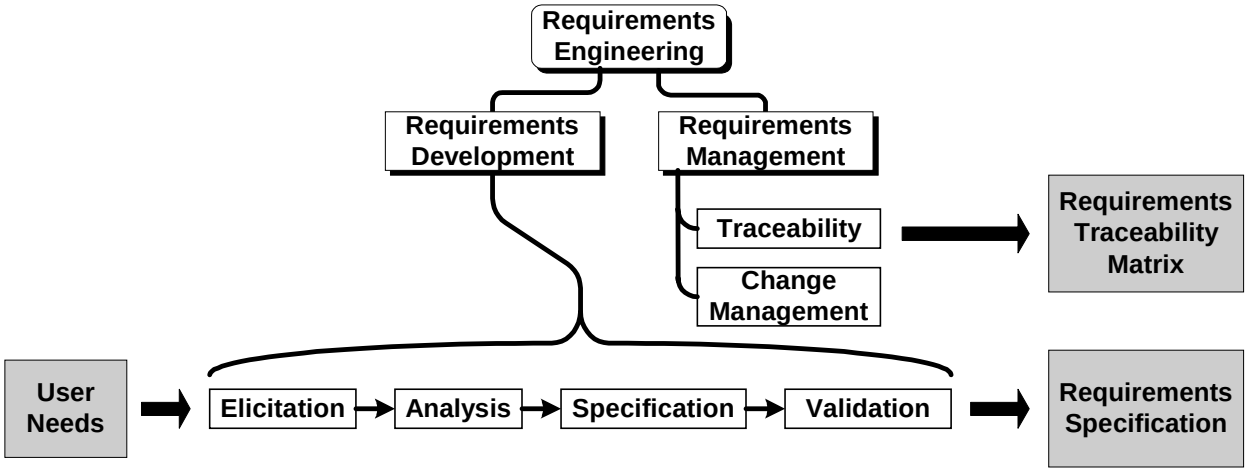

Figure 4-2 Requirements Engineering Process Overview [5]

The process shows user needs being used to generate a Requirements Specification and a Requirements Traceability Matrix (RTM). The four activities of requirements development form a sub-process, while the activities of requirements management do not flow into each other, but are separate. Each of the activities will be discussed greater detail.

4.2.2.1 Elicitation

The elicitation activity consists of gathering information about user needs, primarily from the users themselves. It is a process of helping the users to understand and articulate their requirements so they can make it known to the developers. What is wanted is information about what users are currently working with, why it is inadequate, what their vision of an improved system is, and why. An excellent skill to implement on a grand scale in this area is reflective listening, where you repeat back in your terms what you think you heard, to see if the other party agrees. The general elicitation procedure consists of five steps: [6]

1. Identify relevant requirements sources.
2. Ask them appropriate questions to understand their needs.
3. Look for implications, inconsistencies, and unresolved issues in gathered information.
4. Confirm your understanding of requirements with the users.
5. Synthesize appropriate statements of the requirements.

There are many ways to elicit user requirements, both direct and indirect. The following list presents several methods commonly used. More detail on each of these can be found in the source document, available via the Internet. [7]

- Questionnaire Interviews
- Open-ended Interviews
- Introspection
- Discourse Analysis
- Discussion
- Protocol Analysis
- Brainstorming
- Focus & Application Development Groups
- Surveys

4.2.2.2 Analysis

Requirements analysis is the process of breaking down user requirements into their components and studying these to develop a set of system requirements. The three major goals of this process are: [4]

1. Achieve agreement among developers and customers.
2. Provide a basis for design.
3. Provide a basis for Verification and Validation (V&V)

The analysis process consists of developing a set of high-level requirements describing the functionality of the overall system, followed by a stepwise process of decomposing the higher-level requirements into successively more detailed functional and nonfunctional requirements. This decomposition will probably have several levels, as shown in Figure 4-3. In addition to these levels there will probably be different sets of requirements developed for hardware, software, and the interfaces between them. There will probably also be requirements relating to Human-Computer Interaction (HCI).

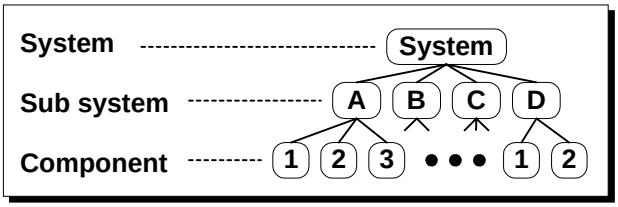

Figure 4-3 Example Requirements Levels

4.2.2.3 Specification

Specification is the documentation of the requirements developed through requirements analysis. This is a critical activity. Requirements must be carefully worded to state clearly what is wanted. Ambiguous, incomplete, or incorrect requirements can have disastrous results in later phases of development. It is essential to specify requirements in such a way that they are not misinterpreted. They should say what you mean and mean what you say. See Section 4.2.3 for more details on the requirement specification.

4.2.2.4 Validation

Requirements must be carefully validated to ensure that they are complete, correct, and unambiguous. Validation should not be confused with verification, which means to determine whether the product meets the requirements. Validation's purpose is to ensure the requirements describe what the user really needs. It involves the users, the developers, and often an independent validating organization. Once requirements are validated through a formal review process, the requirements specification becomes the baseline for all subsequent development and testing activities. In addition to written documents, requirements are usually maintained in a computer database.

4.2.2.5 Traceability

Traceability is a requirements management activity where requirements are traced back to their original higher-level requirement sources. This ensures that all higher-level requirements are being met by detailed requirements. It also ensures that lower-level requirements have a higher-level source and have not been arbitrarily added to the scope of the project. This effort generally employs some type of database tool and produces the requirements traceability matrix. More on the RTM can be found in Section 4.2.4.

4.2.2.6 Change Management

Requirements will change because of technical issues, funding issues, scheduling issues, and other real-life project events. There must be a process in place to deal with these changes. This process defines the requirements management activity and continues on throughout the remainder of the project. The first responsibility of change management is to determine what changes will be allowed. This is usually accomplished by creating a change board con-

sisting of appropriate developers and stakeholders. All change requests must be reviewed by the board, and only those approved by the board are implemented. The second responsibility of change management is to document the changes and make the appropriate updates to the requirements documents and databases. If requirements are properly traced it facilitates tracking changes through the different levels of specifications.

4.2.3 Requirements Specification

4.2.3.1 Contents

The requirements specification is the primary product of the requirements phase, and the directing document for all development and testing efforts. The system will be built or modified, and tested to comply with this document. Figure 4-4 shows the requirements specification broken out into a top-level system definition of specification, and three major subsystem specifications for hardware, software, and the interface between them. Also shown are those development efforts that are directed by the software requirements specification. Although not shown, there would be similar efforts for the hardware and interfaces.

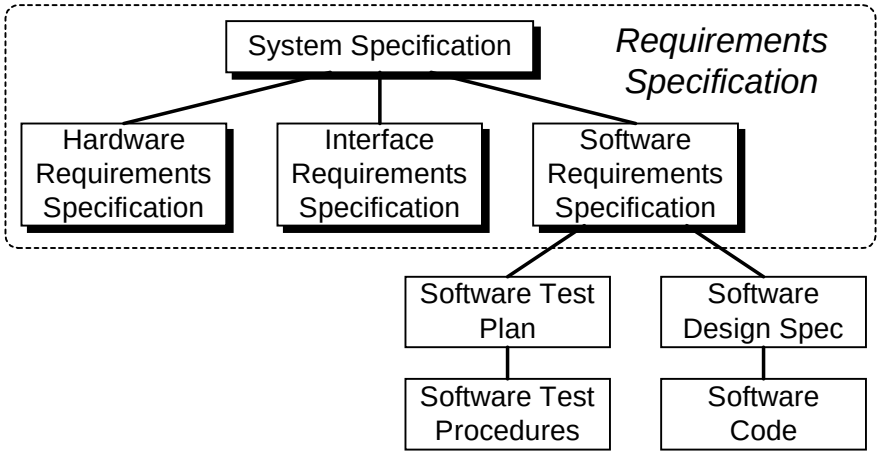

Figure 4-4 Requirements Specification and Follow-on Products

In addition to dividing requirements into different system levels and different disciplines, the requirements specification should contain the following types of requirements and information:

- Functional requirements
  - All inputs to the system.
  - All outputs from the system (data or actions).
  - Information about what must be accomplished between the inputs and outputs (processes within the system.)
  - Method of verification.
- Nonfunctional requirements (reliability, efficiency, maintainability, portability, capacity, etc.) [4]

In software intensive systems these requirements may take the form of natural language, as well as making extensive use of *entity relationship diagrams [8]*, *data flow diagrams [9]*, or other software requirements specification methods.

The requirements specification should not include project requirements, designs, or product assurance plans (Configuration management plan, test plans, validation & verification plans, etc.) [4]

4.2.3.2 Requirements Specification Properties

A good requirements specification should have the following properties: [10]

- Unambiguous
- Consistent
- Usable
- Concise
- Complete
- Modifiable
- Correct
- Feasible
- Verifiable
- Traceable
- Prioritized

4.2.4 Requirements Traceability Matrix

The RTM consists of a database of requirements in a matrix format. Each requirement is numbered and the requirement text is included, along with the source requirement number from higher level documents. When printed out, the RTM becomes a single document with multiple appendices, each appendix containing the trace for a single subsystem specification to a higher specification as shown in Figure 4-5.

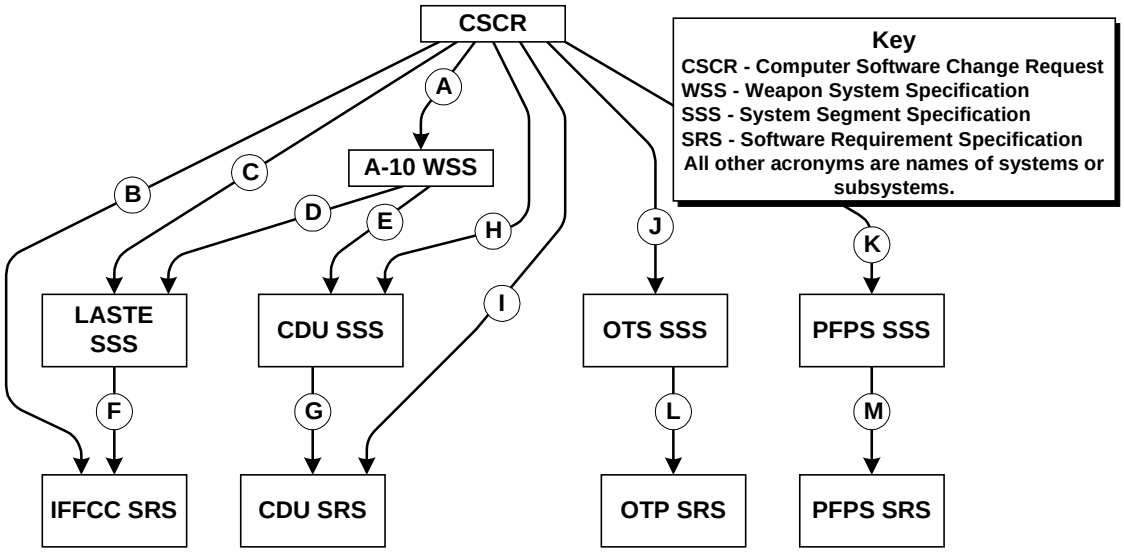

Figure 4-5 Example Requirements Traces Between System Components

In the example shown, the boxes represent different subsystem specifications. The arrows represent the flow down of requirements from higher levels to more detailed levels. The circles contain the letter of the appendix in the RTM which traces the requirements between the two documents.

4.3 Requirements Engineering Checklist

This checklist is provided to assist you in requirements engineering. If you cannot check an item off as affirmative, you need to either rectify the situation or develop a contingency plan to solve problems that may arise.

4.3.1 Requirements Development

- ❑ 1. Have you had extensive user involvement in developing the requirements?
- ❑ 2. Do all stakeholders understand and agree on how the system will be used?
- ❑ 3. Are all stakeholders satisfied with the requirements?
- ❑ 4. Do the developers understand the requirements?
- ❑ 5. Are all requirements clear and unambiguous?
- ❑ 6. Have you distinguished between needs and wants? Are requirements relevant?
- ❑ 7. Are requirements consistent with each other (i.e., they don't conflict.)
- ❑ 8. Are requirements complete? Do the requirements cover everything that is supposed to be accomplished?

- ❑ 9. Has design detail been left out of the requirements?
- ❑ 10. Are all requirements testable?
- ❑ 11. Can you see the requirement as an output?
- ❑ 12. Are all requirements easily recognized as requirements by using the word shall?
- ❑ 13. Have the requirements been prioritized?
- ❑ 14. Are requirements feasible with respect to cost, schedule, and technical capability?
- ❑ 15. Are requirements verifiable?
- ❑ 16. Is the system boundary clearly defined; what is in scope, what is not?
- ❑ 17. Are all external interfaces to the system clearly defined?
- ❑ 18. Is the specification written so that it can be modified when necessary, with minimal impact to the rest of the document?
- ❑ 19. Are you conducting formal and informal reviews of requirements documents?

4.3.2 Requirements Management

- ❑ 20. Have all requirements been entered into the requirements database?
- ❑ 21. Are the requirements traces sorted to allow requirements lookup by paragraph number, requirement number, or other useful index?
- ❑ 22. Can all requirements be traced to original system-level requirements?
- ❑ 23. Are all system-level requirements allocated to lower level, subsystem requirements?
- ❑ 24. Do you have a requirements change process documented and in place?
- ❑ 25. Have you identified members of the requirements change board?
- ❑ 26. Is adequate impact analysis performed for proposed requirements changes?
- ❑ 27. Do you know who is responsible for making the changes?
- ❑ 28. Have requirement changes been traced upward and downward through the higher and lower-level specifications?
- ❑ 29. Do you have a process in place to maintain and control the different versions of the requirements specification?

4.4 Regulations

- AFI 10-601; Mission Needs and Operational Requirements Guidance and Procedures.
- AFPD 10-6; Operational Requirements; Mission Needs and Operational Requirements.
- AFPD 90-11; Planning Systems.
- CJCSI 3170.01A, Requirements Generation System.
- DoD 4120.24-M, Defense Standardization Program (DSP) Policies and Procedures.
- DoD 5000.2-R, Mandatory Procedures for Major Defense Acquisition Programs (MDAPs) and Major Automated Information System (MAIS) Acquisition Programs, Part 2; 2.2, Requirements.
- DoDD 5000.1, The Defense Acquisition System, 4.2.2. Time-Phased Requirements and Communications with Users.
- DoDI 5000.2, Operation of the Defense Acquisition System, 4.6.1.1. Requirements Generation System and E2.1.18. Requirements Authority.
- FAR 11 Describing Agency Needs and FAR 39 Acquisition of Information Resources.

4.5 References

[1] The Standish Group, "The Scope of Software Development Project Failures," 1995.

[2] DoD, *Program Manager's Guide For Managing Software*, v. 0.6, Chapter 6, 21 June 2001.

[3] Gause, Donald and Weinberg, Gerald, *Exploring Requirements: Quality Before Design*, Dorset House, 1989.

[4] Software Technology Support Center Course: *Life Cycle Software Project Management*, Project Initiation, 9 October 2001.

[5] Wiegers, Karl E., "When Telepathy Won't Do: Requirements Engineering Key Practices," Cutter IT Journal, May 2000. www.processimpact.com/articles/telepathy.html

[6] Raghavan, Zelesnik, & Ford, "Lecture Notes of Requirements Elicitation," 1994: www.sei.cmu.edu/publications/documents/ems/94.em.010.html

[7] Henderson, Lisa G.R., "Requirements Elicitation in Open-Source Programs", *Crosstalk*, 2000, www.stsc.hill.af.mil/crosstalk/2000/jul/henderson.asp

[8] SmartDraw, Entity Relationship Diagram tutorial: www.smartdraw.com/resources/centers/software/erd.htm

[9] SmartDraw, Data Flow Diagram tutorial: www.smartdraw.com/resources/centers/software/dfd.htm

[10] Davis, Alan M., *Software Requirements Analysis and Specification*, Prentice-Hall, 1990.

4.6 Resources

Christel & Kang, "Issues in Requirements Elicitation," 1992: www.sei.cmu.edu/publications/documents/92.reports/92.tr.012.html

Crosstalk Magazine: www.stsc.hill.af.mil/crosstalk/

- "Writing Effective Natural Language Requirements Specifications": www.stsc.hill.af.mil/crosstalk/1999/feb/wilson.asp
- "Experiences in the Adoption of Requirements Engineering Technologies": www.stsc.hill.af.mil/crosstalk/1998/dec/cook.asp
- "An Examination of the Effects of Requirements Changes on Software Releases": www.stsc.hill.af.mil/crosstalk/1998/dec/stark.asp
- "Doing Requirements Right the First Time": www.stsc.hill.af.mil/CrossTalk/1998/dec/hammer.asp
- "Four Roads to Use Case Discovery": www.stsc.hill.af.mil/CrossTalk/1998/dec/ham.asp
- "An Examination of the Effects of Requirements Changes on Software Releases": www.stsc.hill.af.mil/CrossTalk/1998/dec/stark.asp
- "Experiences in the Adoption of Requirements Engineering Technologies": www.stsc.hill.af.mil/CrossTalk/1998/dec/cook.asp
- "Recommended Requirements Gathering Practices": www.stsc.hill.af.mil/crosstalk/2002/apr/young.asp
- "Making Requirements Management Work for You": www.stsc.hill.af.mil/crosstalk/1999/apr/davis.asp

Department of Energy (DOE) Software Engineering Methodology, Chapter 4: http://cio.doe.gov/sqse/sem_toc.htm

Department of Energy, Software Risk Management Practical Guide: http://cio.doe.gov/sqse/pm_req.htm

Department of Justice Systems Development Life Cycle Guidance, Chapter 6: www.usdoj.gov/jmd/irm/lifecycle/table.htm

IEEE Computer Society: http://computer.org

- 4th International Conference on Requirements Engineering (ICRE'00): http://computer.org/proceedings/icre/0565/0565toc.htm
- 3rd International Conference on Requirements Engineering (ICRE'98): http://computer.org/proceedings/icre/8356/8356toc.htm
- 2nd International Conference on Requirements Engineering (ICRE '96): http://computer.org/proceedings/icre/7252/7252toc.htm

INCOSE Requirements Working Group: www.incose.org/rwg/

- "Factors Influencing Requirement Management Toolset Selection": www.incose.org/lib/rmtools.html

- "Characteristics of Good Requirements": www.incose.org/rwg/goodreqs.html
- "Requirements Management Technology Overview": www.incose.org/tools/reqsmgmt.html
- "What is a Requirement?": www.incose.org/rwg/what_is.html

Guidelines for the Successful Acquisition and Management of Software-Intensive Systems (GSAM), Version 3.0, Chapter 11, OO-ALC/TISE, May 2000. Available for download at: www.stsc.hill.af.mil/gsam/guid.asp

Heimdahl and Associates, "Requirements Verification Checklist": http://www-users.cs.umn.edu/~heimdahl/csci5802/memos/RequirementsChecklist.pdf

Hooks, Ivy F., "Why Johnny Can't Write Requirements": www.complianceautomation.com/papers/whyjohnny.htm

Leffingwell, Dean A., "A Field Guide to Effective Requirements Management Under SEI's Capability Maturity Model": www.rational.com/products/whitepapers/299.jsp

Ludwig, J., Requirements management site: www.jiludwig.com

Modell, Martin, *A Professional's Guide to Systems Analysis, 2ed.*, full book online: http://www.dai-sho.com/pgsa2/index.html

Program Manager's Guide for Managing Software, 0.6, 29 June 2001, Chapter 6: www.geia.org/sstc/G47/SWMgmtGuide%20Rev%200.4.doc

Rational Software Corp., *UML Notation Guide 1.1*, 1997: http://curie.informatik.fh-luebeck.de/~st/UML/UML1.1/

Requirements Management Place, The: www.rmplace.org

Requirements Generation System, Joint Chiefs of Staff: www.jsc.mil/jsce3/EMCSLSA/stdlib/cd/Added/3170_01b.pdf

Wiegers, Karl E.,

- "10 Requirements Traps to Avoid": www.processimpact.com/articles/reqtraps.html
- "Writing Quality Requirements": www.processimpact.com/articles/qualreqs.html
- "First Things First: Prioritizing Requirements": www.processimpact.com/articles/prioritizing.html
- "Automating Requirements Management": www.processimpact.com/articles/rm_tools.html
- "In Search of Excellent Requirements": http://www.processimpact.com/articles/exc_reqs.html
- "Habits of Effective Analysts": www.processimpact.com/articles/analyst_habits.html