Project Capability and Requirements Guidelines

Date	Rev	Author	Comments
2006-08-15	A	J.M. Conrad	Original Document
2007-02-05	B	J.M. Conrad	Minor change to page 6, 1st paragraph ("about")

Table of Contents

1 Overview

This document describes the steps a Principal Engineer (one of the team members) must follow to create and maintain a project's Capability and Requirements and Acceptance Test Plan documents.

2 Requirement Engineering

Requirements Engineering is the disciplined application of scientific principles and techniques for developing, communicating, and managing requirements [1]. Requirements will serve as the rubric by which Company ABC can verify that the end device has all the functionality the customer desires. To this end, the Principal Engineer (PE) will need to ensure the following tasks are followed.

2.1 Capabilities Gathering

In this step, the PE will be responsible for gathering all information needed to document the capabilities. This includes [1]:

- Identifying relevant published sources of requirements (Statement of Work, Proposals, and BAAs/RFPs).
- Interview/discuss with the customer specific capabilities

2.2 Capabilities Demonstration Test Plan

The Demonstration Test Plan should describe the tests and test steps needed to demonstrate the capabilities of the device. Each test should indicate which capability is being verified. All capabilities must be verified by at least one test.

2.3 Requirements Gathering

In this step, the PE will be responsible for gathering all information needed to document the requirements. This includes [1]:

- Identifying relevant sources of requirements (capabilities, Statement of Work, Proposals, and people).
- Determining what information is needed.
- Analyzing the gathered information, looking for implications, inconsistencies, or unresolved issues.
- Confirming the understanding of the requirements with the source.
- Synthesizing appropriate statements of the requirements.

2.4 Requirements Analysis

The next step will be to create and analyze all requirements. The PE will obtain requirements from all possible sources (include but not limited to) [1]:

- Observation and measurements of the current system.
- Interviews with customers.
- Current system documentation.
- Feasibility studies (Phase I reports).
- Models and prototypes.
- Analytical analysis.

The Requirements Document, once written, must be approved by the customer (Company ABC technical employees and management) and the faculty mentor. Note that some requirements may be ill-defined or vague. These should be noted, and effort should be spent by the PE to investigate the technological area in order to make the requirements more clear BEFORE the design is started.

2.5 Requirements Management

Once the requirement documents have been approved, there should be real proof that a change is necessary with a measurable benefit of the change. The clear responsibility is to keep project within costs, within budget, and to meet customer's needs. Any change will need to go through an analysis to determine the impact (cost and time) of the proposed change. A requirement change must be accepted by the PE, and approved by the customer and the faculty mentor.

2.6 Requirements Verification – Acceptance Test Plan

An Acceptance Test Plan will be written at the same time as the Requirements Document. The Test Plan will represent the steps needed to verify that the requirements have been met. Each test must verify at least one specific requirement. All requirements must be addressed by at least one test. The test plan will consist of specific tests, each with detailed test steps, and each noting which requirement has been addressed.

2.7 Development Steps

The Figure 1 shows the simple steps which the design team will follow for project planning, design, and implementation. The requirements steps described in this document and the documents created are show in the top part of the figure.

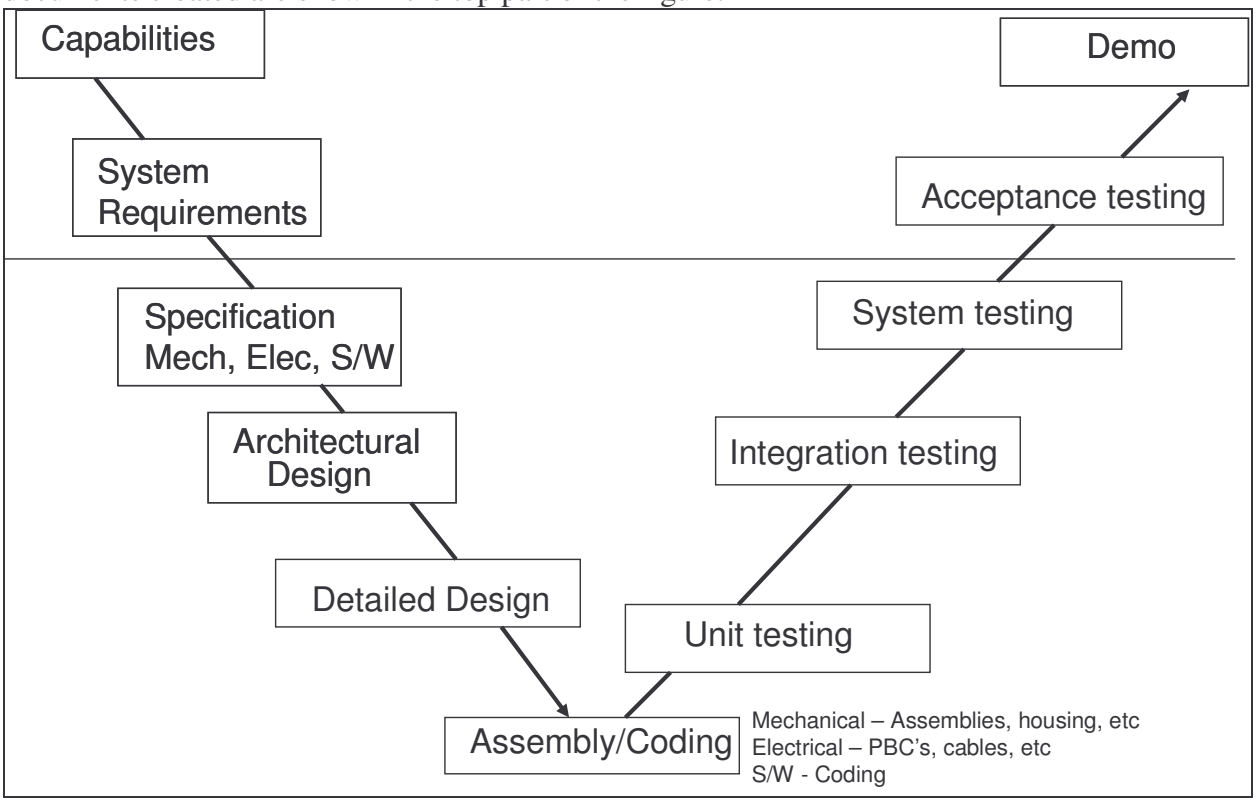

Figure 1: Senior Design Team Development Steps

2.8 Using this Guideline for Requirements Refinement

In situations where the project includes a spiral development cycle (plan1, design1, prototype1, test1, plan2, design2, prototype2, test2, etc.), the requirements should be re-analyzed during each planning phase. The requirements developed in plan1 should form the foundation for the requirements developed in plan2.

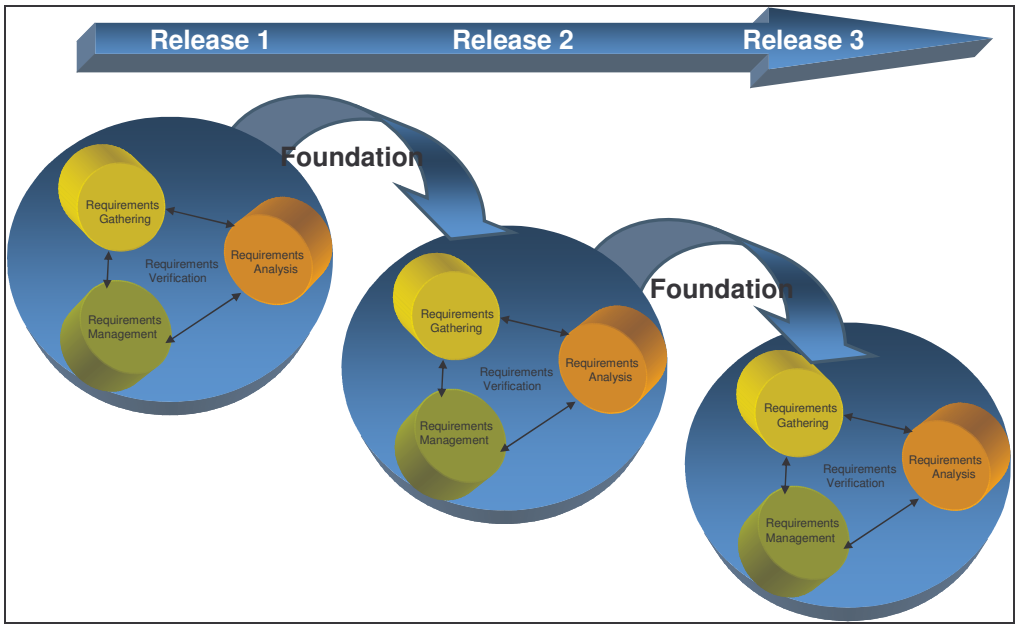

Figure 2: Requirements progressing through multiple prototype development [1]

3 Writing Clear Requirements

From ref. [2]: It is good engineering practice to state the problem in terms of the top-level function that the system must perform. However, it is better to state the problem in terms of the deficiency that must be ameliorated. This stimulates consideration of more alternative designs.

Example 1

Top-level function:	The system shall hold together 2 to 20 pieces of 8½ by 11-inch, 20 pound paper.
Alternatives:	stapler, paper clip, fold the corner, put them in a folder

Example 2

The deficiency:	My reports are typically composed of 2 to 20 pieces of 8½ by 11-inch, 20 pound paper. The pages get out of order and become mixed up with pages of other reports.
Alternatives:	stapler, paper clip, fold the corner, put them in a folder, number the pages, put them in an envelope, throw away the report, convert it to electronic form, have it bound as a book, put it on audio tape, distribute it electronically, put it on a floppy disk, put it on microfiche, transform the written report into a videotape.

3.1 Characteristics of a Good Requirement

This section and list is copied directly from [2]:

1. **Describes What, Not How:** First and foremost, a good requirement defines what a system is to do and to what extent, but does not specify how the system is to do it. A statement of a requirement should not be a preconceived solution to the problem that is to be solved. To avoid this trap, ask why the requirement is needed, then derive the real requirements.
2. **Atomic**: A requirement should be "atomic," not compound. That is, it should have a single purpose (one idea per requirement). Furthermore, each requirement should be allocated to a single physical entity. It is acceptable to assign two or more requirements to one physical component. However, it would be a mistake to assign one requirement to two physical components.
3. **Unique**: A requirement should have a unique label, a unique name, and unique contents. Avoid repeating requirements.
4. **Traceable**: A good requirement is traceable; it should be possible to trace each requirement back to its source (Capability, customer).
5. **Necessary**: All requirements should be necessary. Systems Engineers should ask, "Is this requirement really necessary? Will the system necessarily be better because of this requirement?" Avoid over-specifying the system, writing pages and pages that no one will probably ever read. There are two common types of over-specification: gold plating and specifying unnecessary things. For example, requiring that the outside of a CPU box be gold-plated is not a good requirement because something far less expensive would probably be just as effective. Also, requiring that the inside of the CPU box be painted pink is probably an unnecessary request.
6. **Complete**: The documentation must be as clear, concise, and complete as possible.
7. **Semantic Usage:** Avoid the use of synonyms (e.g., the software requires 8 Mbytes of RAM but 12 Mbytes of memory are recommended) and homonyms (e.g., Summaries of disk X-rays should be stored on disk.
8. **Is Not Always Written:** It must be noted that all systems will undoubtedly have many "common sense" requirements that will not be written. This is acceptable as long as the requirements really are common sense. An exhaustive list of requirements would take years upon years and use reams of paper, and even then you would probably never finish.
9. **Quantitative and Testable:** Quantitative values must be given in requirements. A requirement states a necessary attribute of a system to be designed. The designer cannot design the system if a magnitude is not given for each attribute. Without quantification, system failure could occur because (1) the system exceeded the minimum necessary cost due to over design, or (2) failed to account for a needed capability. Quantitative values for attributes are also necessary in order to test the product to verify that it satisfies its requirements.

   Each requirement must be verifiable by examination, analysis, test, or documentation and therefore must have a well-defined figure of merit. Qualitative words like *low* and *high* shall be (at least roughly) defined. What is low cost to a big corporation and what is low cost to a small company may be very different. Only requirements that are clear and concise will be easily testable. Requirements with ambiguous qualifiers will probably have to be refined before testing will be possible. Furthermore, the value given should be fully described as, for example, an expected value, a median, a minimum, a maximum, etc. A requirement such as

"reliability shall be at least 0.999" is a good requirement because it is testable, quantified, and the value is fully described as a minimum. Also the requirement "the car's gas mileage should be about 30 miles per gallon" is not a good requirement as it establishes an approximate performance measure. It would have been stronger without the word “about”.

Note that often the customer will state a requirement that is not quantified. For example: "The system should be aesthetically pleasing." It is then the engineer's task to define a requirement that is quantified, i.e., "The test for aesthetics will involve polling two hundred potential users; at least 70% should find the system aesthetically pleasing."

It is also important to make the requirements easily testable. NASA once issued a request for proposals for a radio antenna that could withstand earthquakes and high winds. They said, "The antenna shall not deflect by more than 0.5 degrees in spite of 0.5 G forces, 100 knot steady winds or gusts of up to 150 knots." They expected bids around $15 million. But all of their bids were around $30 million. NASA asked the contractors why the bids were so high. The contractors said testing the system was going to be very expensive. NASA revised the requirements to When 'hit with a hammer,' the antenna shall have a resonant frequency less than 0.75 Hz. Then they got bids between $12 and $15 million.

10. **Identifies Applicable States:** Some requirements only apply when the system is in certain states or modes. If the requirement is only to be met sometimes, the requirement statement should reflect when. There may be two requirements that are not intended to be satisfied simultaneously, but they could be at great expense.

    For example: The vehicle shall

    (1) be able to tow a 2,000-pound cargo trailer at highway speed (65 mph),

    (2) accelerate from 0 to 60 mph in less than 9.5 seconds.

    It would be expensive to build a car that satisfied both requirements simultaneously.
11. **States Assumptions:** All assumptions should be stated. Unstated bad assumptions are one cause of bad requirements.
12. **Use of Shall, Should, and Will:** A mandatory requirement should be expressed using the word shall (e.g., the system shall conform to all state laws.). A preference requirement can be expressed using should or may (e.g., the total cost for the car's accessories should be about 10% of the total cost of the car.). The term will can be used to express a declaration of purpose on the part of a contracting agency, to express simple future tense and for statement of fact (e.g., the resistors will be supplied by an outside manufacturer.).
13. **Avoids Certain Words:** The words optimize, maximize, and minimize should not be used in stating requirements, because we could never prove that we had achieved them. Consider the following criteria: (1) we should minimize human suffering, and (2) we should maximize the quality and quantity of human life. A starving child should be fed, even if the child continues to live in misery. However, the criterion of minimal suffering could lead to the conclusion that the child should die.

    Requirements should not use the word simultaneous because it means different things to different people. It might mean within a few fempto seconds to a physicist, on the same clock cycle to a computer engineer, or to an anthropologist studying the extinction of the dinosaurs, within the same millennium.
14. **Might Vary in Level of Detail:** The amount of detail in the requirements depends upon the intended supplier. For in-house work or work to be done by a supplier with well-established systems engineering procedures, the requirements can be written at a high level. However,

for outside contractors with unknown systems engineering capabilities, the requirements might be broken down to a very fine level of detail.

15. **States its Rationale:** Although it is seldom done, it would be nice if each requirement stated why it was written and what it was supposed to ensure.

3.2 Types of Requirements

The Requirements Document will have the following sections:

1. Introduction of the project/product
2. Functional Requirements: specify specific behaviors of a system. They define the internal workings of the system: that is, the calculations, technical details, sizes, data manipulation and processing, and other specific functionality.
3. Non-functional Requirements: specify criteria that can be used to judge the operation of a system, rather than specific behaviors.
4. Constraints - limits the development in some way, such as, defining an operating system the system must run on, which programming language must be used to implement the system, what drawing tools must be used, or specific vendors which must be used for manufacture.
5. Acceptance Test Plan - steps needed to verify that the requirements have been met

4 Capability and Requirements Steps

Requirements development should use the following steps. These are to be done by the Project Engineer (or someone assigned by the Project Engineer).

1. Gather all of the documents associated with the project: Proposals, emails, SOWs, BAA/RFP. (Capabilities and Requirements Gathering)
2. Sit down with the customer and interview them about what they think the capabilities of the end-product should be. (Capabilities Gathering)
3. Write the **Capability document** that includes as many capabilities you can find from the sources above. (Capabilities Analysis)
4. Conduct a review meeting with the customer and the faculty mentor. The result of the meeting should be a version of the **Capability document** that all agree to and will sign. (Capabilities Analysis)
5. Gather all of the documents associated with previous projects: Requirements, lessons-learned, final reports. (Requirements Gathering)
6. Sit down with customer and the faculty mentor and interview them about what they think the requirements of the end-product should be, and of any other requirements not described in the written sources. (Requirements Gathering)
7. Write the **Requirements document** that includes as many requirements you can find from the three sources above. (Requirements Analysis)
8. Pull together the project team for a brainstorming session (1-2 hours). Have them read the existing **Capability and Requirements documents**, and develop additional requirements that may have been missed. (Requirements Analysis)
9. Complete the **Requirements document**. (Requirements Analysis)

10. Conduct a review meeting with the customer and the faculty mentor. The result of the meeting should be a version of the **Requirements document** that all agree to and will sign. (Requirements Analysis)
11. Complete the **Requirements document** and write the **Acceptance Test Plan** that addresses all requirements. (Requirements Analysis and Verification)
12. Obtain signatures for both of these documents from the management.
13. If a requirement is poorly understood, you will need to conduct further investigations on the technology. This may include prototyping. (Requirements Management)
14. During the rest of the project, if anyone wants to add or change a requirement:
    a. Fill out the **Requirements Change Impact Analysis document**.
    b. Conduct a change review with the customer and the faculty mentor present (this can also be done via email).
    c. Update the **Requirements document**.

5 References

1. Madigan, M, "Requirements Engineering", University of Colorado ECEN 5543 Course Notes, 2002.
2. Bahill, A.T. and Dean, F., Discovering System Requirements, Chapter 4 in the *Handbook of Systems Engineering and Management,* A.P. Sage and W.B. Rouse (Eds), John Wiley & Sons, 175-220, 1999.