Successful Strategies To Improve Your Requirements

NDIA 2005 Systems Engineering Conference
October 27, 2005

Timothy G. Olson, President
Quality Improvement Consultants, Inc.
(760) 804-1405
Tim.Olson@qic-inc.com
www.qic-inc.com

Objectives

Describe some requirements problems from industry.

Present a useful classification of requirements problems.

Describe some practical strategies and best practices that organizations have used to successfully develop and manage their requirements.

Provide real examples that address requirements problems.

Answer any of your questions.
Why Focus on Requirements?

“The hardest single part of building a software system is deciding what to build... No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.”

Fredrick Brooks, Jr. [Brooks 87]
Why Focus on Requirements?

One study [Beichter 84] estimates that 60% of system errors are due to inadequate specification and design.

According to the SEI National Software Capacity Study [SEI 90], the top 2 out of 10 factors that contribute to the failure of system development contracts to meet schedule or costs are requirements problems (1 to 5 scale: 1=not serious; 3=serious; 5=very serious):

1. Inadequate requirements specification (4.5)
2. Changes in requirements (4.3)

Why Focus on Requirements?

A recent research report from the Standish Group highlighted the continuing quality and delivery problems in our industry and identified three leading causes:

- Lack of user input
- Incomplete requirements and specifications
- Changing requirement specifications

Outline

Why Focus on Requirements?

A Practical Requirements Classification

CMMI\textsuperscript{SM} Requirements Overview

Practical Approaches for Requirements

Requirement Examples

Some Advanced Approaches

Summary

Problems with Requirements

According to the SEI [Christel 92], problems of requirements elicitation can be grouped into 3 categories:

1. **Problems of Scope**: the requirements may address too little or too much information.

2. **Problems of Understanding**: problems within groups as well as between groups such as users and developers.

3. **Problems of Volatility**: the changing nature of requirements.
Scope and Volatility

The list of 10 requirements elicitation problems given in [McDermid 89] can be classified according to the 3 categories in [Christel 92]:

Problems of Scope
- The boundary of the system is ill-defined
- Unnecessary design information may be given

Problems of Volatility
- Requirements evolve over time

Problems of Understanding
- 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”
Outline

Why Focus on Requirements?

A Practical Requirements Classification

CMMI<sup>SM</sup> Requirements Overview

Practical Approaches for Requirements

Requirement Examples

Some Advanced Approaches

Summary

Requirements Management (REQM)

SG 1: Manage Requirements:

SP 1.1-1: Obtain an Understanding of the Requirements

SP 1.2-2: Obtain Commitment to Requirements

SP 1.3-1: Manage Requirements Changes

SP 1.4-2: Maintain Bidirectional Traceability of Requirements

SP 1.5-1: Identify Inconsistencies between Project Work and Requirements

• Reference: "Capability Maturity Model<sup>®</sup> Integration (CMMI<sup>®</sup>), Version 1.1", CMU/SEI-2002-TR-011, March 2002
**World-Class Quality**

### Requirements Development (RD)

**SG 1: Develop Customer Requirements:**
- SP 1.1-1: Collect Stakeholder Needs
- SP 1.1-2: Elicit Needs
- SP 1.2-1: Develop the Customer Requirements

**SG 2: Develop Product Requirements:**
- SP 2.1-1: Establish Product and Product-Component Requirements
- SP 2.2-1: Allocate Product-Component Requirements
- SP 2.3-1: Identify Interface Requirements

**SG 3: Analyze and Verify Requirements:**
- SP 3.1-1: Establish Operational Concepts and Scenarios
- SP 3.2-1: Establish a Definition of Required Functionality
- SP 3.3-1: Analyze Requirements
- SP 3.4-3: Analyze Requirements to Achieve Balance
- SP 3.5-1: Validate Requirements
- SP 3.5-2: Validate Requirements with Comprehensive Methods


---

### Engineering Process Areas

- **RD**: Requirements Development
- **TS**: Technical Solutions
- **PI**: Project Integration
- **VER**: Verification
- **VAL**: Validation

CMMI<sup>SM</sup> and Requirements

Requirement processes need to be defined, trained, and improved (e.g., OPF, OPD, OT, OID).

Support processes are critical for measuring and managing requirements (e.g., CM, MA, PPQA).

Defects need to be removed and prevented in requirements (e.g., PI, VER, VAL, CAR).

IPPD also contains allocating requirements to teams (e.g., IPM for IPPD).

Supplier Sourcing requires managing supplier requirements.

Outline

Why Focus on Requirements?

A Practical Requirements Classification

CMMI<sup>SM</sup> Requirements Overview

Practical Approaches for Requirements

Requirement Examples

Some Advanced Approaches

Summary
Practical Strategies

1. Use the CMMI<sup>SM</sup> or SW-CMM®: Requirements Management (REQM or RM)
2. Use Configuration Management (CM)
3. Use requirements metrics (e.g., priority, stability, risk, number of requirements, defect density, etc).
4. Define the requirements process, and use lessons learned from quality (e.g., QFD, Juran, etc).
5. Tailor a requirements standard (e.g., IEEE).
6. Use inspections and defect prevention.
7. Use operational definitions to define requirements.

Outline

Why Focus on Requirements?
A Practical Requirements Classification
CMMI<sup>SM</sup> Requirements Overview
Practical Approaches for Requirements

Requirement Examples

Some Advanced Approaches

Summary
1. Requirements Management Process

- Customer Requirements
  → Develop Requirements

- Change Requests
- Problem Reports
  → Manage Requirements

- Product Requirements
  → Updated Product Requirements

2. Use CM

Fundamental Baselines

Requirements Baseline → Implementation Baseline → Product Baseline

Place the requirements under formal CM and use CCB’s to control changes.
3. Example Requirement Metrics

<table>
<thead>
<tr>
<th>#</th>
<th>Requirement</th>
<th>Reference (e.g., customer)</th>
<th>Allocation</th>
<th>Stability (H/M/L)</th>
<th>Risk (H/M/L)</th>
<th>Priority (H/M/L)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>System shall send an RTF FAX</td>
<td>SOW # 10-20.3</td>
<td>5 - 1 Second</td>
<td>H</td>
<td>L</td>
<td>M</td>
</tr>
<tr>
<td>2</td>
<td>Aircraft position shall be updated by the Inertial Navigation System (INS) Solution</td>
<td>ORD #2-30-20.3.4.4</td>
<td>100 - 50 milli-Second</td>
<td>M</td>
<td>M</td>
<td>H</td>
</tr>
</tbody>
</table>

4. Documentation Framework

- **POLICIES**
  - “Laws” or “Principles” that govern operations
  - Guide/Govern
  - Processes
  - “What happens over time” to build products
  - Implemented by
  - Procedures
  - “How to” or step by step instructions

- **STANDARDS**
  - “Operational definitions” & “acceptance criteria”

- **TRAINING**
  - Provides the needed knowledge and skills

- **TOOLS**
  - Supports and automates operations

*Slide adapted from “A Software Process Framework for the SEI Capability Maturity Model”, CMU/SEI-94-HB-01*
4. Requirements Process - NASA Onboard Shuttle Project

- Requirements conception
  - Identify need
  - Examine architectural options
  - Develop software system solution

- Requirements generation
  - Define software requirements in accordance with operational concept and system requirements
  - Produce requirements specification

- Requirements analysis
  - Assess technical and resource impact
  - Determine acceptability, implementability, testability
  - Examine requirements readiness

- Requirements inspection
  - Discuss proposed requirement in detail
  - Discuss operational scenarios
  - Identify issues and errors

- Requirements approval
  - Evaluate risks and benefits
  - Decide on resource expenditures
  - Establish baseline

- iteration
  - Correct errors
  - Resolve issues
  - Rewrite

4. Manage Requirements Process

**Purpose:** Effectively Manage Req. Changes

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Entry</th>
<th>Tasks</th>
<th>eXit</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Req.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change Requests</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem Reports</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Roles:** Project Manager (PM), CCB
5. IEEE SyRS/SRS Standard

IEEE System or Software Requirements Standard:

1.0 Introduction
2.0 Overall Description
3.0 Specific Requirements
   3.1 External Interface Requirements
   3.2 Functional Requirements
   3.3 Performance Requirements
   3.4 Design Constraints
   3.5 System Attributes
   3.6 Other Requirements
4.0 Traceability Matrix
5.0 Appendices

• Reference: “IEEE Recommended Practice for Software Requirements Specifications”, IEEE Std 830-1993

6. Example Requirements Checklist Categories

1. Clarity
2. Completeness
3. Complexity
4. Consistency
5. Constraints
6. Feasibility
7. Functionality/Logic
8. Interfaces
9. Standards
10. TBDs
11. Testability
12. Traceability
Etc.
7. Example Operational Definition

What is a good requirement? When is a requirement defined? Questions like these are difficult to answer without operational definitions.

An operational definition precisely and concisely defines a measurable requirement that states [NASA 96]:

- What it has to do
- How well it has to do it
- Under what conditions it has to do it

<table>
<thead>
<tr>
<th>#</th>
<th>Requirement (What)</th>
<th>Conditions</th>
<th>Upper Limit</th>
<th>Lower limit</th>
<th>Base Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Report total percentage of students that passed the first test and graduated</td>
<td>Students that pass first test by (\geq 70%) score</td>
<td>Calculate Percentage to 3 decimal places</td>
<td>Plus or minus .001</td>
<td>Percent</td>
</tr>
<tr>
<td>2</td>
<td>Report total percentage of students that failed the second test and did not graduate</td>
<td>Students that failed second test by &lt; a 70% score</td>
<td>Calculate Percentage to 3 decimal places</td>
<td>Plus or minus .001</td>
<td>Percent</td>
</tr>
</tbody>
</table>
Outline

Why Focus on Requirements?

A Practical Requirements Classification

CMMI<sup>SM</sup> Requirements Overview

Practical Approaches for Requirements

Requirement Examples

Some Advanced Approaches

Summary

Some Advanced Strategies

**Juran Model:** Customer requirements are written in the customer’s language, then translated into the product requirements written in producer’s language.

**QFD/Juran’s Quality Planning Process:** Measurable requirements that meet customer needs using a defined process.

**Usage Scenarios/Use Cases/Operational Scenarios:** A powerful way to identify requirements based on user needs.

**Requirements written in formal languages.**
Outline

Why Focus on Requirements?

A Practical Requirements Classification

CMMI\textsuperscript{SM} Requirements Overview

Practical Approaches for Requirements

Requirement Examples

Some Advanced Approaches

Summary

Summary

The hardest single part of building a software system is the requirements.

The top requirements problems in software are inadequate requirements specifications, changes to requirements, and lack of user input.

Requirements elicitation problems fall into problems of scope, understanding, and volatility.

There are practical strategies that you can use today that will help you address problems with requirements.
References

- [Olson 03a] Olson, Timothy G. “Measurably Improving Your Requirements Based on the CMMIVSM”, NDIA CMMI Conference, Denver, CO, 2002.
- [Olson 03b] Olson, Timothy G. “Measurably Improving Your Requirements Based on the CMMIVSM”, NDIA CMMI Conference, Denver, CO, 2002.