Department of Energy Quality Managers
Software Quality Assurance Subcommittee
Reference Document SQAS19.01.00 - 2000

Guidelines for Requirements Management

April 2000

United States Department of Energy

Albuquerque Operations Office

Abstract

Requirements management is a vital and important component of project management. This guideline is intended to help project teams and organizations plan, manage, and improve the requirements for their systems. This guideline supports the requirements management key process area discussed in Level 2 of the Software Engineering Institute's (SEI) Software Capability Maturity Model (CMM-SW). It also provides information for implementing and improving requirements management as indicated in Level 3, Organizational Process Definition and Integrated Software Management.

Acknowledgement

This document and an accompanying promotional trifold brochure were prepared for the Department of Energy (DOE) by a Working Group of the DOE Quality Managers' Software Quality Assurance Subcommittee (SQAS). At the time this document was prepared, the Working Group and other major contributors were:

Andy Bicocchi	**DOE/CIO**
Mike Blackledge	**SA**
Kathy Burris	**LA**
Kathleen Canal	**DOE/CIO**
Brenda Coblentz	**DOE/CIO, Chair**
Ray Cullen	**SR**
Gary Echert	**DOE/AL**
Orval Hart	**LA, Scribe**
Sara Kidwell	**DOE/CIO**
Cathy Kuhn	**KC**
John Mashburn	**OR**
Jonathan Parker	**LA**
David Peercy	**SA**
Ed Russell	**LL**
Nancy Storch	**LL**
Ellis Sykes	**KC**
Patty Trellue	**SA**
Bill Warren	**LL**
Graham Wing	**AWE**

Table of Contents

Figures

Tables

Preface

Purpose. The intent of this guideline is to improve requirements management capabilities. This is an area of information systems engineering which is vital to the successful completion and maintainability of a product. It has been identified by the Software Quality Assurance Subcommittee (SQAS) as needing significant attention because requirements are difficult to manage even with today's automated tools.

Audience. Because requirements management is a critical partnership between customers, managers, and practitioners, this document is targeted to all three.

- For *customers*, requirements management provides documentation of what should be delivered, which may also serve as the contractual basis for system development. Requirements management captures, verifies and validates users' needs.

- For *managers*, requirements management provides the basis for scheduling and measuring progress and addressing organizational concerns.

- For *practitioners* (e.g., designers, coders, quality assurance staff), requirements management provides the specifications for design, and the basis for validating, verifying, and testing the product.

Scope. The main focus of this document is on software projects. However, the scope is not limited to the software, but includes the total environment in which the software operates. It applies to all software systems - business, scientific, and technical. Additionally, requirements management should be implemented for existing systems, as well as new systems.

Document Overview. This document provides guidelines for establishing, implementing, and sustaining requirements management processes and conducting requirements management for a product through four key activities. It consists of this preface, which discusses the purpose, audience, scope, and overview of the document, four chapters, and two appendices.

Chapter 1, Requirements Management Overview, provides an overview of requirements management; i.e., what it is, why it is important, and general information.

Chapter 2, Requirements Management, describes requirements management as both product and project lifecycle processes. It discusses how formal requirements management processes are implemented and sustained as a lifecycle activity.

Chapter 3, Requirements Management in Practice, discusses the four major activities of the requirements management process - gathering, documenting, verifying, and managing changes.

Chapter 4, Requirements Management Process Improvement, discusses improving an organization's requirements management processes.

Appendix A provides a list of definitions, acronyms, and abbreviations used in the document.

Appendix B contains references and bibliography information.

1. Requirements Management Overview

1.1 Introduction

Whatever the nature of a development, enhancement or maintenance project, it is initiated because there is a need or mission, and that need or mission is decomposed into requirements. Therefore, the objective(s) of a project is the implementation of those requirements to meet the need or mission. Tracking requirements throughout the lifecycle of a project ensures they are met and provides evidence that the project team met their objective or mission. Requirements management does not stop with the end of a project but continues into the operation and support of the product.

1.1.1 What is Requirements Management?

Requirements management, in system/software engineering, is the process of controlling the identification, allocation, and flowdown of requirements from the system level to the module or part level, including interfaces, verification, modifications, and status monitoring. [IEEE BP07738]

Requirements management is the set of activities that concentrate on assuring the specifications, i.e., requirements, are met to the customer's satisfaction. It is a process that begins at project inception and continues until the resulting product(s) is no longer needed. Requirements management includes the major requirements management phases, such as organizing, implementing and sustaining and the key requirements management activities, such as gathering, documenting, verifying and managing changes.

There are both commercially available and in-house developed tools that automate some of the requirements management processes. The use of tools is encouraged to improve both productivity and quality.

1.1.2 Why is Requirements Management Important?

Requirements provide the scope and direction for projects. When requirements are not managed well, a project can fail or be very costly. This has been the history in industry and government, and the prospects are not bright. The following results of a 1994 IBM survey of 24 leading software companies illustrate the importance of requirements management.

- 2 out of every 8 large systems development efforts started will be canceled.
- 75% of projects delivered will have operating failures (do not function as intended or not used at all).
- The average software development project takes 50% longer than planned.
- 55% of projects cost more than expected.

Requirements errors (defects) found at the requirements stage cost only about one-fifth of what they would cost if found at the testing stage and one-fifteenth of what they would cost after the system is in use. [KONTONYA]. Why is this so? If an error is detected when writing the specification, only the specification needs to be corrected. If an error is detected during design,

then the design and the specification have to be corrected, and the process goes on. Costs to fix an error at a later stage are also greater because it is more difficult to understand the cause of the error and to correctly formulate a solution to it.

Requirements are very vulnerable to changes external and internal to the project, many of which are not within the control of the project team. The more complex the project, the greater the vulnerability. The Gartner Group in an April 15, 1997, article entitled "Requirements Management: Taming Scope Scourge" prophetically states that:

> *Through 2000, more than 50 percent of large applications development projects that lack project managers trained in--and authorized for--scope management will either be canceled or consume at least twice their initial budgeted resources (0.8 probability). [GARTNER]*

From industry reports, this appears to be true. The paradigm needs to change. It begins with implementing and improving requirements management.

1.1.3 Who's Responsible?

All of the participants in a project - customers, managers, and practitioners - are responsible for requirements management. Specific roles are typically identified for each project; however, all participants provide varying degrees of support to requirements management activities.

1.2 Critical Role of Requirements Throughout the Lifecycle

A product's lifecycle is driven by the requirements; consequently, requirements must be reassessed throughout the lifecycle. In particular, requirements must be traceable throughout design and test because the success of a project revolves around requirements that are complete, accurate, and measurable.

2. Requirements Management

The desired end results of requirements management are:

- Delivery of a product that meets the stated needs and expectations, is manageable, and requires minimal changes or rework.
- Implementation of the expected product with no or very few customer complaints.
- Basis for controlling the budget, schedule, resources, and quality practices.
- Development of an agreement between customers, managers, and practitioners on the project and the technical basis for acceptance of the software.

An information system or an application for an information system is considered a product. A product follows a product lifecycle from inception through development to maintenance and ultimately retirement. The set of activities associated with developing the first and subsequent product releases are projects. There may be many projects that develop the initial release of, and subsequently make changes to, the product over the product's lifecycle. The initial and subsequent releases are developed based on identified requirements. Requirements management entails activities associated with both capturing and maintaining the requirements for the product's overall lifecycle, and the allocating and tracking of requirements associated with projects.

2.1 Requirements Management - A Product Lifecycle Process

There are three major phases in the management of requirements throughout a product's lifecycle: Organizing, Implementing, and Sustaining. Some organizations may have established requirements management processes defining these phases for use across all products (see Appendix A for definition of "organization"). Others may leave it up to each project team to define these phases. At a minimum, all projects should have processes for phases defined in the Project Plan or a Requirements Management Plan to ensure the product's requirements are managed throughout the product's lifecycle. The relationships of the three phases are illustrated in Figure 2-1.

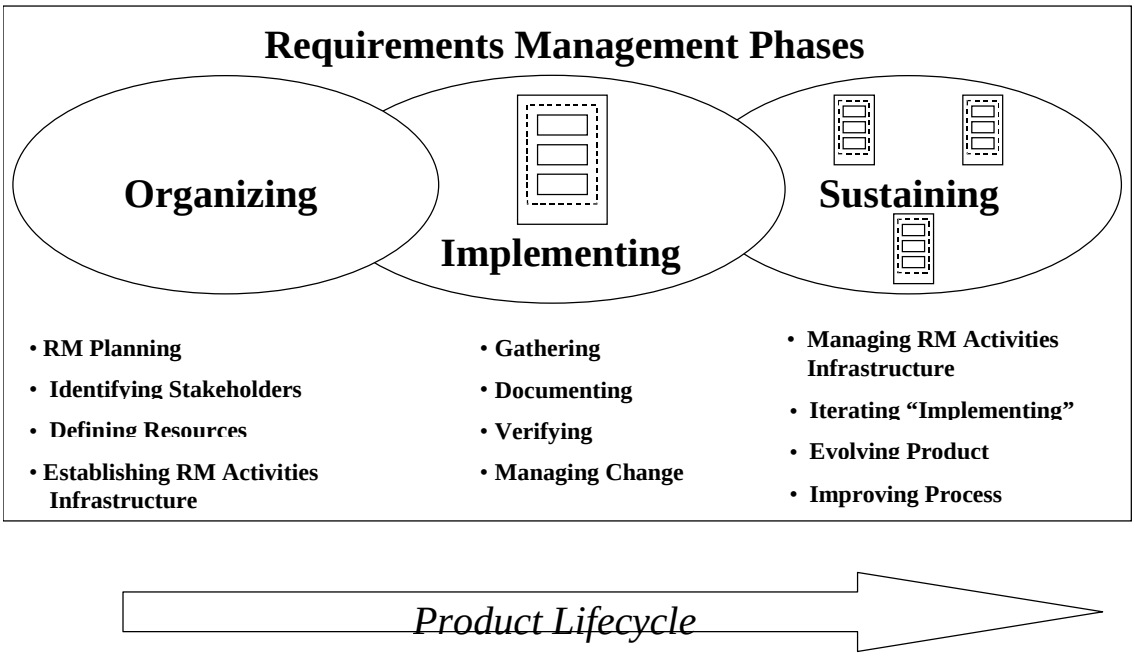

Figure 2-1. Relationship of Requirements Management Phases

As can be seen, the requirements management phases overlap due to the dynamic nature of the product lifecycle. After the Implementing phase begins and there is an understanding of the need and requirements for the product, it may be necessary to make changes to decisions made in the Organizing phase, such as additional resources, new tasks or approach, or additional tools. Also, in the Implementing phase, the processes for managing change to the work products begin, then the managing change activity is continued in the Sustaining phase. In the Sustaining phase, after the baseline product is produced, changes to requirements can cause further modifications to the decisions made in the Organizing and Implementing phases.

Also, it may be necessary to iterate among the requirements management phases to ensure requirements are adequately defined, implemented, and controlled, and improvement can occur. In other words, the requirements management phases not only overlap but also are cyclical throughout the product's lifecycle. It is not a step-by-step process. Sections 2.1.1 through 2.1.3 describe the activities that occur in each of the requirements management phases in more detail.

2.1.1 Organizing

Organizing requirements management should be addressed early in a product's lifecycle. Table 2-1 shows the major components of this phase.

Table 2-1. Components of the Organizing Phase of Requirements Management	
Inputs:	• External Project/Product Requirements • Internal Resource Availability and Capability
Resources:	• Customer, supplier management, supplier engineering, users and all other stakeholders and affected functional areas
Process:	• Identify project stakeholders and sources of product requirements • Identify general or high-level product technical requirements, constraints, and management methods • Identify general or high-level product non-technical requirements, constraints, and management methods • Identify personnel skills for requirements management • Identify personnel availability and skills match for project and product lifecycle requirements management • Identify Project Manager and Quality Assurance Representative to foster an integrated requirements management approach
Controls:	• Project contractual information • Customer and supplier standards, policies, and guidelines • Metrics, analysis, assessments, and reporting for tracking requirements management goals
Outputs:	• Infrastructure for implementing requirements management integrated within the project, project management and quality assurance organizations • Requirements Management Plan or inclusion of requirements management planning in Project Plan • Goals and Metrics

Project success will depend upon how well the stakeholders agree on what the product's requirements are, both non-technical and technical. Non-technical requirements, such as cost, schedule, and resources, as well as technical requirements drive tradeoffs throughout the product and project lifecycles. The Organizing phase can be adapted to address the organization's mission and business needs and, therefore, the project's needs.

The resulting deliverable or work product from the Organizing phase is a Requirements Management Plan, or the inclusion of requirements management planning in the Project Plan. Requirements management should be tracked against goals using metrics established in the Organizing phase and metrics data tracked throughout the Implementing and Sustaining phases.

2.1.2 Implementing

Once the infrastructure for requirements management is established, the activities of the Implementing phase can begin. Table 2-2 shows the major components of this phase.

Table 2-2. Components of the Implementing Phase of Requirements Management	
Inputs:	• Infrastructure for implementing requirements management integrated within the project, project management and quality assurance organizations (i.e., this is the output from the Organizing phase) • Requirements Management Plan or inclusion of requirements management planning in Project Plan • Goals and Metrics
Resources:	• Tools, customer, supplier management, supplier engineering, users, and all other stakeholders and affected functional areas
Process:	• Gather requirements from customer • Document requirements in various representation and specification forms • Verify requirements in the specification (and throughout the project and product lifecycle; i.e., design, implementation, test, acceptance, and verification information) • Manage changes to requirements (e.g., change requests, representation and specification forms through formal configuration management process)
Controls:	• Project contractual information • Customer and supplier standards, policies, and guidelines • Metrics and analysis, reviews and assessments, and reporting for tracking user delivery, installation, support constraints, etc.
Outputs:	• Requirements management products (e.g., cost/schedule status, requirements specifications, verification/analysis information, customer acceptance, change control process and procedures and requirements traceability matrix) • Deliverable products (application products, requirements support products)

Within the Implementing phase there are four major requirements management activities - gathering, documenting, verifying, and managing changes to requirements. Chapter 2.2, Requirements Management – A Project Lifecycle Process, describes these four activities in more detail. The major deliverables from the Implementing phase are the specification document(s) and traceability matrix.

2.1.3 Sustaining

The Sustaining phase is essentially a repetitive and iterative evolution of the Organizing and Implementing phases as a result of changes to the requirements of the product. The Sustaining phase can begin during project development or later in the operation and support of the product. Basically, it is a set of change control procedures that enact a review and possible revision to the Organizing phase or repetition of the Implementing phase. For example, changes to the project or product can impact the Organizing phase if additional or fewer resources are required. Also, changes can impact the Implementing phase by causing a revision to the documentation resulting from the gathering, documenting, and verifying activities. Table 2-3 shows the major components in this phase.

Table 2-3. Components of the Sustaining Phase of Requirements Management	
Inputs:	• Baselined or revised requirements management products (e.g., cost/schedule status, requirements specifications, verification/analysis information, customer acceptance, change control process and procedures and requirements traceability matrix) • Initial or revised deliverable products (application products, requirements support products)
Resources:	• Tools, customer, support management, support engineering, users, and all other stakeholders and affected functional areas
Process:	• Change analysis: analyze change request for requirements impact • Change implementation: gather/document requirements changes in representations and specifications forms • Change verification: verify changes in specifications, design, implementation, tests, acceptance • Change release: formalize and document changes in appropriate work products, e.g., specifications, test plans, design plans, and continue change management tracking
Controls:	• Support project contractual information • Customer and supporter standards, policies, and guidelines • Metrics and analysis, reviews and assessments, and reporting for tracking user delivery, installation, support constraints
Outputs:	• Updated versions of requirements management products (e.g., specifications, verification/analysis information, customer acceptance, change control procedures) • Updated versions of deliverable products (e.g., application products, requirements support products) • Updated requirements management processes

Change control procedures should be established during the project lifecycle so that there are disciplined procedures for managing changes during product development. These change control procedures should be continued when the product is turned over to support staff (i.e., operation and maintenance). Project goals and metrics established in the Organizing phase should be verified before turning the product over to support staff. These goals should be affirmed and tracked throughout the lifecycle of the product.

2.2 Requirements Management - A Project Lifecycle Process

The three major phases as illustrated in Figure 2-1 are essentially the infrastructure that governs the requirements management activities for the entire product lifecycle. Each product release is developed as a separate project, or can be thought of as a separate project, which typically follows a lifecycle process. Throughout the project lifecycle, requirements management activities are conducted. As product requirements are allocated to projects, the requirements management activities of the Implementing Phase are conducted and/or iterated.

The requirements management lifecycle process (as discussed in section 2.1 or defined by an organization) should be adaptable and flexible to meet project needs. The scope and implementation characteristics of the requirements management lifecycle process on a project will vary depending on several key factors:

- Project size and complexity
- Project personnel experience

- Customer personnel experience
- Application domain
- Purpose/usage of the application

The purpose or goal of requirements management at the project level is to establish a common understanding between the customer and the provider to ensure the customer's requirements are addressed by the project. Requirements management provides a method for:

- Developing a complete, validated set of requirements including functions, interfaces, and performances for the software or system.
- Dealing with complexity caused by interactions of software and system components and the constraints placed on both.
- Dealing with modifications of the requirements and communicating with project participants.
- Dealing with inadequate allocation of requirements due to non-technical issues (e.g., resources).

Therefore, a requirements management lifecycle process should provide assurance for requirements management at the project level that:

- Responsibilities for requirement identification, traceability, documentation, and testing are understood.
- Each requirement is correctly identified.
- Standards for requirement format and content statement guidelines are available.
- Provides traceability through the lifecycle stages and includes well-defined entry and exit criteria.
- Requirements are easily traced from the source to allocated components and back.
- An overall process is in place to manage changes to requirements, including appropriate automated tool support for the bookkeeping and configuration management activities.

Figure 2-2 illustrates the activities of the Implementing phase. The four major requirements management activities of a project lifecycle (within the Implementing phase) are gathering, documenting, verifying and managing change. These activities are dynamic and may be repeated throughout the project lifecycle. The managing change activity is an overarching activity that interrelates to the other three activities. Also, as requirements are gathered and verified, the requirements documenting activities continually occur. It may be that these requirements management activities for system, software, and hardware requirements are all at different points in the Implementing phase, but at some time they are synchronized.

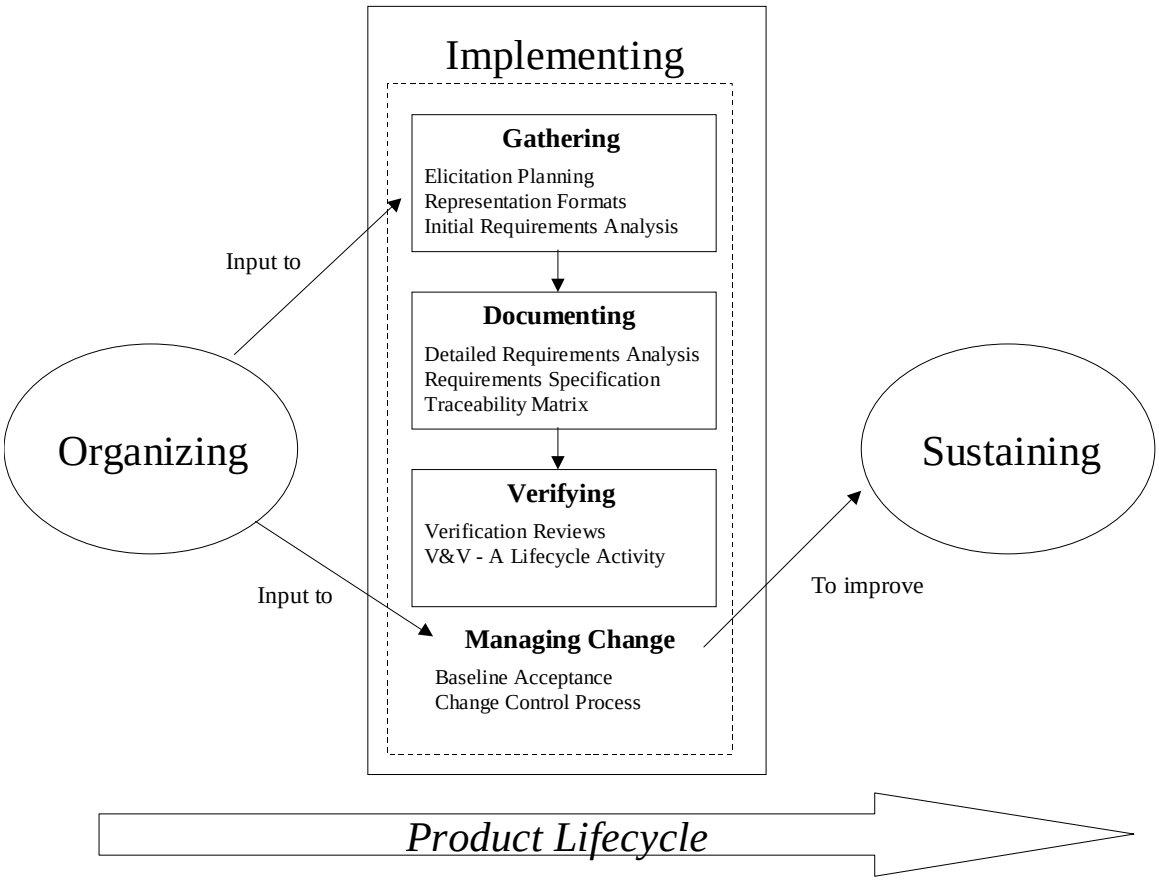

Figure 2-2. Implementing Phase

An overview of the four activities is summarized in Table 2-4. A description of how these activities are applied in practice is provided in Chapter 3, Requirements Management in Practice.

Table 2-4. Four Major Requirements Management Activities

ACTIVITIES	DESCRIPTION
Gathering	Gathering and documenting requirements is an iterative communication activity between customers, managers, and practitioners (project stakeholders) to discover, define, refine, and record a precise representation of product requirements. Various methods are used for gathering requirements. Some initial analyses, such as grouping, categorizing, prioritizing, are performed during this activity. A more detailed analysis is performed in the Documenting activity.
Documenting	After the requirements have been gathered, they are analyzed in detail and documented in a requirements specification. The resulting requirements specification and any derived component hardware/software requirements specifications serve as a record of customer agreement and supplier commitment. These specifications are tracked using a Requirements Traceability Matrix and are subject to verification and change management throughout the product lifecycle.
Verifying	Once the requirements specification has been developed, the requirements are verified. Verifying requirements is a process to ensure that the product requirements specification is an accurate representation of the customer's needs. This process also ensures that the requirements are traced and verified through the various lifecycle phases; particularly design, implementation and testing. Requirements must be traceable from external sources such as the customer, to derived system level requirements, to specific hardware/software product requirements. In addition, all of these requirements must be cross-traceable to design, implementation, and test artifacts to ensure requirements have been satisfied.
Managing Changes	After the requirements, as documented in the requirements specification, have been verified and agreed upon, they can be baselined and the last task, managing changes, begins. Managing changes is a formal process to identify, evaluate, track, and report proposed and approved changes to the product specification. As a project evolves, requirements may change or expand to accommodate modifications in project scope or design. Advances in technology and the project team's increased insight can also affect requirements. These changes to the requirements can impact the project cost, resources, and schedule. A change control process provides an accurate and complete trail of all changes that are pertinent to the project.

3. Requirements Management in Practice

Requirements management is a time consuming activity and should be performed by those with appropriate training and/or experience. Before engaging in the process, the following should be considered:

Good requirements management is more likely to occur if best practices are followed. Some best practices include:

- The project team understands and follows an appropriate lifecycle.
- Customers, users, and all important stakeholders are involved.
- Good communication occurs during the process of deriving requirements.
- Difficulties that stem from the inherent complexity of the product or the computing environment are addressed early on.
- A prototype or early version of the software system in addition to a specification document is developed so that requirements can be better defined before they are baselined. (*Note:* These should be considered as tools for conducting requirements analysis and should not serve as the design; i.e., they simulate but do not actually perform the operations.)
- A list of requirements is maintained in a database where they are more easily managed.
- A requirements configuration management process is implemented.
- Requirements are not added or changed without (1) conducting a risk analysis to determine impact on project plans, and (2) re-estimating the cost and schedule of the project.
- Some operational capabilities are delivered during the development process with the intention to add the remaining or new requirements.

Care must be taken not to state an implementation when specifying requirements. For example, "develop a database" is an implementation for design; "provide the ability to sort" is a requirement, which may need the implementation of a database. Stating the implementation in a requirements specification may cause a design that would not be the best solution or cause other requirements to be overlooked. One solution to determining whether the requirement is truly a requirement or an implementation is to ask "why" a requirement is needed.

Requirements need to be grouped, their relationships identified, and integrated before integrated test and evaluation begins. This not only helps determine whether the requirements grouping will satisfy the customer's need, but also validates requirements at acceptance. Thus, there is a need for test plans to reference the requirements and the methods for demonstrating compliance.

A method for requirements traceability should be utilized throughout the product's lifecycle. This is not an easy process and it is suggested that an automated tool be used.

The requirements management model (i.e., the way one conducts the process) that is used depends on the specific non-technical requirements of the project and the selected development lifecycle. For example, if the iterative build model is used for the development lifecycle, then requirements would be baselined at the initial prototype and the requirements management process would be repeated as requirements are added to the prototype. Also, a successful

requirements management model should provide a mechanism for conducting requirements management activities in parallel and for going backwards in the process as defects in previously defined and documented requirements are identified.

While managing requirements, it is important to avoid scope creep. If scope creep appears to be occurring, yet is justified, (e.g., customer realizes the problem is too narrow and needs to increase scope), then the process may need to be reviewed and the project schedule adjusted.

3.1 Gathering Requirements

In the Gathering activity, the project stakeholders work together to discover, define, refine, and record a precise representation of product requirements. Gathering includes planning the elicitation, determining the representative formats, and conducting an initial analysis. Involved in these tasks is discovering requirements through interaction with customers; reviewing the current technology infrastructure; and refining, modeling, and specifying precise requirements representation. It is a complex activity that is prone to error, which can be minimized through the usage of techniques, tools, and experience. Also, planning decisions regarding the requirements analysis activities need to be known at the beginning of the Gathering activity since those decisions impact the tasks conducted for this activity.

The objectives of the Gathering activity are to:

- Include customers and users as full participants in the requirements identification process.
- Identify stakeholder interfaces (i.e., customer, management, engineering, and users).
- Establish communication mechanisms (i.e., on-site visits, written memos, how information is going to be exchanged).
- Initiate the Gathering activity (i.e., communicate, capture, analyze, feedback, agree, iterate).
- Collect software-related technical, non-technical, and acceptance criteria requirements.
- Define software requirements including inputs, processing, and outputs.
- Define product requirements for the specification.

3.1.1 Elicitation Planning

Planning for the requirements elicitation includes defining the types of questions needed to get an overview of the project and for defining the functions, constraints, and assumptions. No matter how well prepared, ambiguities and missing requirements can occur, sometimes resulting in a conflict. Therefore, establishing communications is very important not only for requirements elicitation but also for conflict resolution. Also, decisions will need to be made for how to describe and qualify requirements.

3.1.2 Representation Formats

Decisions need to be made on the representation formats for communicating requirements. Requirements may be captured in various representation forms. Usually Gathering and Documenting involves some iteration among the various agents and representation forms prior to allocation of requirements to components and specification documentation.

3.1.3 Initial Requirements Analysis

Some analysis is performed in the Gathering activity in preparation for the more detailed analysis performed in the Documenting activity. Requirements analysis is the set of activities from which requirements specifications (i.e., documentation) derive. At this point, analysis may only be to identify and clarify the attributes or characteristics, attach the requirements to functions, categorize, prioritize, and allocate to objects/classes, and define the constraints. Standard requirements analysis methods such as checklists and automated tools should be available to ensure that a cost-effective and efficient job of analysis is possible. These tasks help when the more detailed requirements analysis is performed.

For the detailed analysis of a given project, one methodology should be selected and used. The analysis method should have been selected in planning and definitely should be selected before detailed analysis. The decision should not wait until the detailed analysis begins. Different methodologies are available as approaches for requirements analysis. In following the analysis approach and principles selected for the project, some of the activities that may be conducted during the Gathering activity and revisited during detailed analysis include:

- Review of existing system specification, product project plan, etc. to determine the basic problem elements as perceived by the customer/user.
- Problem evaluation and solution synthesis - which includes definition of all externally observable data objects, evaluation of the flow and content of information, etc. *Note:* The primary focus of this type of analysis is on “what” not “how”.
- Creation of system architecture modules to better understand data and control flow, functional processing, behavioral operation, and information content.
- Development of the requirements specification, which clearly documents the requirements.
- Development of the requirements traceability matrix to assist in tracking requirements of the product throughout its lifecycle.

The format of the requirements specifications and matrix should be known during the Gathering activity to assist in the data collection scope.

3.1.4 Specifying Requirements

Problems in specifying requirements are created from the project management process and human organization. It is not easy to spot problems or see that current methods are not working. However, established procedures can assist in systematically identifying needs, both existing and anticipated, and for forecasting problems. Some common problems with specifying requirements are as follows:

- Incorrect requirements because the wrong customer is being addressed, articulation difficulties, or not involving relevant users.

- Imprecise and ambiguous requirements due to the nature of human language, deliberate imprecision for flexibility, human conflict preventing consensus, nebulous information-age projects, customer's lack of expertise, or oversights on the part of project planners.

- Shifting requirements due to the need to cut expenses, avoid obstacles, capitalize on opportunities, or the project's progress stimulates new possibilities.

- Over specification of requirements because of insufficient information; initiative, creativity, or quality work is discouraged; requirements are ignored; or flexibility is not allowed.

- Excessive flexibility in specifying requirements because of a lack of or no cohesiveness in the project or the vision as evidenced by patchwork deliverables, chaotic project planning, and time and cost overruns.

There are various methods, techniques, and tools for specifying requirements. Some general guidelines that can be applied by the practitioner are as follows:

- State the requirements explicitly and have the project staff and customers sign off on them. Requirements are usually implied and need to be written down.

- Assume that if a requirement can be misinterpreted, it will be misinterpreted. Ask for interpretations of requirements.

- Recognize that there will be changes on your project and that things will not go precisely as anticipated. Avoid rigidity and anticipate change.

- To as great an extent as possible, include pictures, graphs, physical models, and other nonverbal exhibits in the formulation of requirements.

- Establish a system to monitor carefully any changes made to the requirements. Address how changes will affect the whole project and document the cost.

- Educate project staff and customers to the problems of specifying requirements, especially the requirements lifecycle. The quality of the project plan is tied to the quality of requirements, and changes in requirements impact the project's budget.

3.2 Documenting Requirements

Documenting is the organization of precise requirements from the Gathering activity into a product, the requirements specification, which is used for communication throughout the project. There can be several requirements specifications depending upon the scope and size of the software product. Once developed, the requirements specification document is subject to verification and change management throughout the project's lifecycle.

Also, the traceability matrix (see section 3.2.4) will be initiated or updated during the Documenting activity. Some traceability matrices are initiated in the Gathering activity and thus

provide information on why (e.g., no technology to implement, lack of funding, and out of scope) some requirements are not considered for the product or this project.

Documenting requirements is an iterative process that crosscuts the other three requirements management activities of Gathering, Verifying, and Managing Changes. For example, some of the artifacts from the Gathering activity may be refined and incorporated into the requirements specification. These requirements will be verified and perhaps later changed, which subsequently will cause the requirements specification to be changed.

The objectives for the Documenting activity are to:

- Establish the requirements specification artifacts (i.e., format for the overall specification and product requirements information).
- Identify representation method(s) (i.e., how is information going to be captured and represented for purposes of analysis and easy transition to specification documentation, models to be used; e.g., Quality Function Deployment (QFD)).
- Capture requirements information in representation forms (e.g., possibly use the representation artifacts from the Gathering activity for analysis).
- Complete a requirements traceability matrix (before the requirements specification is documented).
- Transition representation forms/information into a requirements specification (i.e., move information from its gathering representation format to its documentation specification format).

3.2.1 Representation Formats

Decisions made during the Gathering activity on the representation formats for communicating requirements are applied to Documenting. Determining useful representation mechanisms for the requirements and an organization strategy for grouping the requirements is a particularly important part of the requirements documentation activity. Representation formats for incorporation into the requirements specification may include those used for gathering the requirements and other formats; e.g., such as simple text, formal requirements language, context, and others.

3.2.2 Detailed Requirements Analysis

An in-depth analysis of the requirements is conducted in the Documenting activity using the analysis approach and principles selected for the project and any initial analysis documentation. Requirements analysis is the process of studying user needs to arrive at a definition of the requirements. Several products can result or be refined from the analysis process. These products include a concept of operations, system requirements specification, software requirements specification, and software design specification, to name a few.

An initial analysis should be conducted to eliminate non-essential information. The detailed analysis activity should move from essential information toward implementation detail. In particular, requirements allocated to software should focus on the software products that are needed and the functions that are to be performed. Activities to satisfy the Documenting

objectives mentioned previously are conducted. In addition, the requirements should be verified against the quality attributes selected for the product.

There are levels of analysis; e.g., the customer and perhaps supplier will have requirements analysis from which the initial "customer requirements" result. This is usually accomplished in the Gathering activity. Then, the supplier conducts further detailed requirements analysis, perhaps with various prototyping activities. The supplier may return to the customer to determine if the initial "gathering" is still correct. This may result in an iteration of the Gathering activity and further requirements analysis; i.e., the backward and forward nature of requirements management.

The customer should be involved in the analysis process. One way to do this is to develop an expectation list of what the customer hopes to gain from the product or system. Ask the customer which part of the new or enhanced product or system will be most valuable and find out why. The answers will help to discover what is really expected. Once that is done, compare the project resources with the expectations and negotiate priorities with the customer. For example, the expectations can be put into one of three categories - do now, do later, or not possible. Categorizing can also lead to solutions and customer satisfaction, and reveal constraints on the product design and development process. While doing the analysis, the analyst should keep in mind the following questions to ensure the project objectives will be satisfied:

- What is the problem to be solved and in what context?
- What must the solution do (i.e., functionality)?
- How well must the solution do it (i.e., performance)? Within what constraints? Within what interfaces?
- What level of risk is acceptable (i.e., as a product or system design philosophy)?
- How will the best option be selected from candidate solutions?
- How will it be proven that the solution meets the requirements (verification and validation planning)?
- What documentation is required?

Depending upon the scope and nature of the project, the resolution to these questions may need a review of the problem by focusing on one or more of the following areas: (1) the process; i.e., how the system uses inputs to produce outputs, (2) data-entity relationships, (3) control; i.e., synchronization, deadlock, exclusion, concurrence, process activation/deactivation, etc., or (4) the classes of objects; e.g., in the case of object-oriented coding.

Also, for large projects, at some point the requirements will be refined to where they can be assigned to specific requirements teams, e.g., software, hardware, or subsystem. These teams can then continue to decompose the requirements so that they are sufficiently detailed to be implemented.

3.2.3 Requirements Specification

The requirements specification is a documented and organized interpretation of all the information collected from the interviews, analysis of existing documentation (including the

work breakdown structure), and other elicitation activities which are refined through the requirements analysis process. It is a baseline of the project and is subject to verification and change management throughout the project. Developing clear and complete requirements before full-scale design and code, and practicing good requirements management is important because this leads to:

- A mutual understanding between customers and project teams about the product or system requirements.
- An agreed-upon and approved product requirements specification that becomes the initial baseline for product design.
- Approved communications between the customer and project managers.
- Approved project management and cost control.
- Smooth transition into the product or system design activities and product integrity during coding.
- Achievement of desired functionality and performance for the operation of the product.
- Improved testability and maintainability.
- Improved traceability of test cases to requirements.
- Improved product, system, and user interfaces.
- Improved customer satisfaction and overall quality and reliability.

The collection of the information into a formal requirements specification document occurs initially when enough information is available to begin some form of configuration control. One challenge for the requirements management staff is whether the information that is received is based on opinion or actual experience or knowledge.

Once approved, the specification is then ready to be verified. In cases where there are non-concurrences, solutions and actions should be offered to overcome the issues and concerns. Non-concurrences can be tabled and resolved during the verification process. After the initial document is created, reviewed, verified, and approved, it is put under configuration management in order to control the iterative changes that occur over the rest of the product's lifecycle.

3.2.4 Traceability Matrix

A traceability matrix is a verification tool to trace a requirement throughout the lifecycle. It should be developed because it provides visibility into completeness of the quantitative definition and testability of each requirement. A graded approach may be embedded into this matrix to clarify the relative importance of the items to the overall quality of the system. Automated techniques may be utilized to identify the linkages from requirements to design or testing.

Traceability is the tracking of the disposition of a requirement from its inception in the Gathering activity to its corresponding design, to the code implemented to meet the design and requirements, and to the test case(s) necessary for validating the requirement, and finally to implementation (i.e., throughout the lifecycle). Traceability shows that user needs are met and assures that the product or system will not work in unintended ways. Traceability provides (1) the how and why the product satisfies the stated requirements and (2) information on testing, performance measures, nonfunctional characteristics, and so forth. It is needed for change

control, development process control, and risk control. Traceability is also key in establishing auditability of the system during development and maintaining it after the system is operational.

Requirements traceability is a labor-intensive activity to define the elements that trace to each other. The reward is avoided errors or rework. It is important to know the source of all requirements so they can be verified that they are necessary, accurate, and complete. Requirements must be traceable forward through the lifecycle and backward through the lifecycle.

Traceability of product items must be eventually linked to actual test cases. Each requirement must be traceable to a software function and each software function must be traceable to a user requirement. Also, each requirement should be identified with a specific project objective described in the Project Plan. This identification assures that the product will satisfy all requirements and will not include inappropriate or extraneous functionality.

3.3 Verifying Requirements

During the requirements management process, the requirements must be verified to ensure they are an accurate representation of the customer's needs. This activity must encompass the overall approach of the project, as well as pertinent details, such as flow and content of information, product functions and behavior, interface characteristics, and other design constraints.

Verification links or provides traceability between requirements and design, especially the system and architectural designs. It shows the degree of product quality and compliance with system functional and nonfunctional requirements. Product (e.g., software) verification checks that the allocation of system requirements to the product is appropriate and correct and determines how well the product requirements have been specified in regards to their attributes. It also must ensure that requirements are carried and traced through the various lifecycle phases; e.g., design, implementation, and testing.

Verification is often used with and sometimes thought to be synonymous with validation. Also, verification is sometimes defined as traceability and validation is sometimes defined as testing. Both verification and validation (V&V) will be addressed in this section to show their differences and interaction.

Together, V&V are the activities for determining if (1) requirements are complete and correct, (2) the products developed during the lifecycle fulfill the requirements, and (3) the product complies with the requirements specification. Verification answers the question from the "inside"; i.e., "are we building the system right". Validation answers the question from the "outside"; i.e., "are we building the right system". What would be considered "inside" and "outside" is relative to who you (customer, manager, practitioner, etc.) are and how you are looking at the system (e.g., user versus developer viewpoint).

The objectives of verification typically are:

- Review the feasibility study or concept of operations to determine if project objectives have been met.

- Inspect requirements specifications (via formal requirements specification inspection, walkthrough, peer review, or other inspections).
- Analyze the design against the specification(s) (via formal design inspection, walkthrough, peer review, trace analysis, or other analyses).
- Verify the implementation of the product against the specifications (via formal source code inspection, walkthrough, peer review, trace analysis, unit/integration/system test, or other verification).
- Inspect and verify the traceability matrix.

In contrast to verification, validation is analyses and tests that are conducted on the product during development to prove that the product performs its intended functions correctly, performs no unintended functions, and provides information on quality and reliability. Validation is concerned with the overall product.

The objectives of validation typically are:

- Provide evidence that implementation satisfies the customer's requirements (via demonstration, customer acceptance test, or other operational tests).
- Trace each customer requirement to one or more validation test.
- Trace each validation test to one or more customer requirement.
- Ensure system interfaces are adequately tested.
- Support planning of the system test infrastructure.
- Document results of validation activities.

Although the requirements management process is concerned with verifying requirements throughout the lifecycle, V&V is addressed in the requirements stage of the product's lifecycle. During the requirements stage, the V&V Test Plan (that should have been initiated in the project planning stage) is evolved to include functional and nonfunctional requirements (i.e., performance, external interfaces, design constraints, and attributes). Some of the tools and techniques used for V&V are requirements traceability analysis, requirements evaluation, requirements interface analysis, test planning, control flow analysis, data flow analysis, algorithm analysis, simulation analysis, in-process audits, and requirements walkthroughs.

3.3.1 Verification - A Lifecycle Activity

Requirements are verified (and validated) throughout a product's lifecycle; i.e., mainly during the requirements, design, test, product implementation, and acceptance processes. Therefore, it is important that the verification activity shows requirements are consistent and traceable across the various requirements elicitation and analyses, design, test, and implementation artifacts. The following brief discussion of requirements activities at each of these lifecycle stages illustrates the challenge for verification.

Requirements Definition. Requirements are baselined when the requirements specification(s) is approved by the customer in the Documenting activity. The document(s) then undergoes the first verification activity and lifecycle verification begins. However, requirements may not be complete until the end of the product implementation. Therefore, renegotiations of the requirements should be a part of baselining the requirements specification. The configuration

management process should be in place at the time the baseline is established since requirements can change and, therefore, must be under configuration management.

Design. The requirements are expanded into a full implementation approach during the design stage. One consideration in planning requirements management is the selection of a requirements analysis technique and automated tools that are compatible with the design technique. This helps assure the requirements can be verified in design. Sometimes the difference between requirements and design is confused. Requirements is the what; design is the how. Those documenting requirements write design assumptions. Those designing the product based on the requirements write design decisions. Requirements should drive design. Also, although the two are distinct, requirements analysis and design are interrelated. The output of requirements analysis is input to design, and the output of design is input to requirements analysis. In other words, requirements management verifies the software design; i.e., requirements should be traceable through design and forward and backward between the software requirements to the design.

Test. Testing is necessary to validate that the project and product requirements have been met. The requirements should be mapped according to which requirements are verified in what test phase (unit, integration, system, and acceptance). To ensure requirements have been properly implemented, a set of test activities (static and dynamic) is designed including appropriate trace information from requirements to test activities and from test activities back to requirements. The set of test activities includes all forms of verification and validation. Static tests include activities to verify requirements have been captured in design, allocated to implemented software modules, and traced to one or more test cases (unit, integrated, system). Dynamic test activities can include simulation, partial, or full execution of some or all components and verification/validation of solutions with analytically known solutions or experimental data. In all cases, the test activities are designed to show, through well-defined traceability information and test results, that requirements have been satisfied or exceeded.

Requirements need to be traced to test documents. It is a best practice to initiate test documents in the project planning stage. Then, the test plan can be further developed in the requirements definition stage. There can be several types of test documents (e.g., design, unit, integration) and organization of the related information (e.g., test design, cases, procedures, results logs, incidents, trace matrices). Each document should have traceability information that indicates which requirements are being tested and by which tests. Test documents are prepared from the requirements specification and may be formal or informal. If the requirements specification has errors of omission, incompleteness, or incorrectness, this will be discovered by the test developers early in the lifecycle especially if an independent group (i.e., not the developers) is developing the test documents.

Implementation. During product implementation, the requirements traceability matrix may be expanded. All work products developed during the code, unit testing, and build processes must be traced back to the project requirements and product or system design and verified.

Acceptance. Acceptance testing of the product should verify and validate that (1) all of the requirements have been met and that unintended functions or products were not created, and (2) to provide information about its quality and reliability.

To summarize, any change to requirements during the lifecycle needs to be documented and verified in the requirements specification and traceability matrix. Also, requirements measurements that provide evidence and visibility of satisfaction of the requirements should be presented, when requested, during the project. The requirements management artifacts and traceability products help to provide the substantiation of this evidence.

3.3.2 Verification Reviews

Initially and during the product's lifecycle, the requirements specification is verified that it (1) meets the standards for a requirements specification, (2) is an accurate representation of gathered requirements, (3) records both the customer agreement and supplier commitment via their signatures, and (4) is baselined and under configuration management. These goals are assured through verification reviews, generally called peer reviews.

Peer reviews take several forms, such as formal reviews, structured walkthroughs, and inspections (see Appendix A for definitions). Peer reviews are intended to remove defects from the requirements, the by-products, and the final product through a review process. What is reviewed and how depends on the purpose and objectivity of the review. Reference [IEEE1028] provides more information about the different review types.

In general, reviews are usually performed by peers of the project's team to provide a more independent viewpoint. Some reviews are conducted by people who are involved in the project or created the work product. Typically technical experts are needed to adequately review technical issues in order to ensure the requirements are well defined and correctly implemented. Some activities that may be performed during reviews are:

- Inspect each documented requirement for:
  - Quality attributes
  - Inconsistencies among requirements
  - Redundancies among requirements
  - Unintended changes to previous baselines
- Identify the impact of derived requirements on the operational concept/scenario.
- Verify that candidate technical requirements have been incorporated in the traceability work product.
- Verify that all formalized product requirements are traceable to a higher-level customer requirement.
- Verify that all higher-level customer requirements are traceable to a formalized product requirement(s).
- Document and (depending on the review method) resolve deficiencies.
- Place review reports under configuration management.

It is recommended that reviews be conducted on the requirements specification, design specification, source code, and test plan software artifacts.

3.4 Managing Requirements Changes

Requirements specification is a dynamic process. Because customer requirements for the product are changeable throughout the product's lifecycle, requirements are not often complete until the end of the product implementation. Therefore, procedures need to be in place for making these changes, especially to the requirements specification and traceability matrix.

Objectives for managing changes to requirements are:

- Establish a change reporting and tracking system.
- Identify proposed and approved changes to the requirements specification.
- Evaluate proposed and approved changes to the requirements specification.
- Track proposed and approved changes to the requirements specification.
- Report proposed and approved changes to the requirements specification.

3.4.1 Change Control Process

A formal change control process is used to identify, evaluate, track, and report proposed and approved changes to the requirements, even past the product implementation and into operation and maintenance. In a change control process, approved changes are incorporated into the requirements specification in such a way as to provide an accurate and complete audit trail of the changes.

Sometimes changes to the requirements can adversely affect the project cost, resources, and schedule which is important to the change release. Therefore, changes need to be scheduled and negotiated. Additionally, the requirements traceability matrix and other configuration management procedures are used to manage changes to requirements.

Additional information about configuration management can be found in [SQAS-CM] and in [IEEE828] and [IEEE1024].

3.4.2 Baseline Acceptance

After the requirements documents have been verified, approved, and committed to by the customer and stakeholders, requirements are baselined, and any changes to the requirements must be managed under change control procedures established in the Software Configuration Management Plan. This plan should include the activities for change analysis, change implementation, change verification, change release.

Besides the requirements specification, other requirements artifacts may be placed under configuration management such as paper and digital documentation, databases, models, prototypes, computer-aided software engineering tool outputs. These artifacts should be placed on a traceability work product.

4. Requirements Management Process Improvement

Requirements management processes applied to the product and project lifecycles should be improved throughout the lifecycle by applying lessons learned and adapting or modifying the requirements management processes in each phase; i.e., Organizing, Implementing, and Sustaining. Figure 4-1 below depicts this.

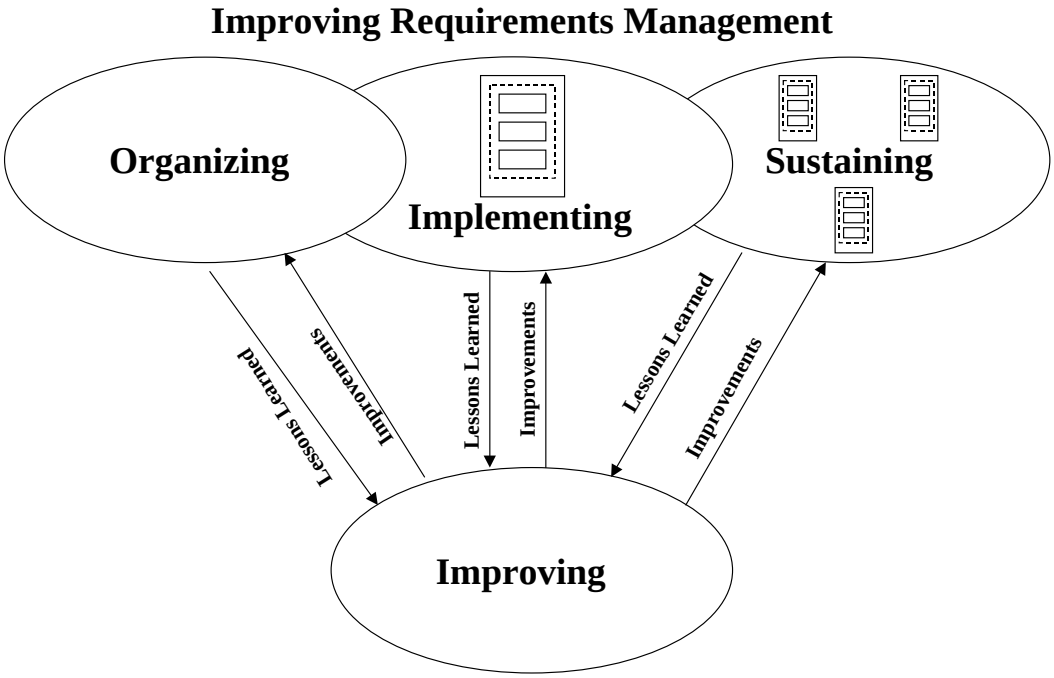

Figure 4-1 Improving Requirements Management

4.1 Requirements Management - An Organizational Focus

The goals of requirements management within an organization should be to provide consistency, predictability, and repeatability of the activities so that improvement can be made. Guidance should be provided for controlling costs, prioritizing requirements, and standardizing requirement analyses methods.

An organization's requirements management processes address the organization's policies and practices of requirements management. They are established to ensure requirements can be adequately engineered throughout the project's and product's lifecycle within the context of the defined project management activities. The organizational standards and guidelines for requirements management should be applied to each project and product so that lessons can be learned and improvements in requirements management can be made. Some organizations establish a Software Process Improvement Group responsible for developing and tracking

improvements to the organization's software processes, which would include requirements management.

4.2 A Process Improvement Model

Requirements management is both a project and organizational responsibility (see Appendix A for definition of "organization"). This distinction is reflected in the Software Engineering Institute's (SEI) Software Capability Maturity Model (CMM-SW) Level 2 and Level 3 guidance. At Level 2, SEI states:

> *"Requirements management establishes a common understanding between the customer and the software project of the customer's requirements that will be addressed by the software project. Requirements management involves establishing and maintaining an agreement with the customer on the requirements for the software project. The agreement covers both the technical and non-technical (e.g., delivery dates) requirements. The agreement forms the basis for estimating, planning, performing, and tracking the software project activities throughout the lifecycle. Whenever the system requirements allocated to the [product] change, the affected [product] plans, work products, and activities are adjusted to remain consistent with the updated requirements." (Quoted from the Software Engineering Institute (SEI) Software Capability Maturity Model (CMM-SW)* [SEICMM]

The SEI CMM-SW provides a model for software capability maturity, i.e., process improvement. Process improvement is a progressive set of activities that moves projects and organizations from *ad hoc* (Level 1) to Level 2 and beyond. Requirements management is a component of that model.

SEI CMM-SW Level 2, Repeatable, provides a framework for identifying, planning, scheduling, costing, verifying, tracing, testing, evaluating, changing, and renegotiating requirements to ensure the customer receives the correct product. Being Level 2 involves knowing the overall approach and details of the project such as the flow and content of information, product functions and behavior, interface characteristics, and other design and test constraints throughout the various lifecycle phases.

At SEI CMM-SW Level 3, Defined, the organization will be engaged in improving requirements management and, thereby, improving the management and success of a project. At Level 3, requirements management is standardized and is consistently applied across the organization. The Level 3 key process areas Organizational Process Focus, Organizational Process Definition, and Software Product Engineering include the process of requirements management within their scope.

4.3 Measurements and Metrics

Measurements and metrics should be established by the organization and project teams for requirements management processes. Measurements are analyzed to track and report status and progress on a project and to improve requirements management processes. Some basic metrics include measurements for tracking (1) the number of changes proposed and approved; (2) budget allocations for added requirements and new skills needed on a project due to changes; and, (3) the effectiveness of requirements management process improvements.

Appendix A: Definitions, Acronyms, and Abbreviations

A.1 Definitions and Terminology

Generic software terminology is defined in [IEEE610] and [SQAS-GLOSSARY], and "The Capability Maturity Model Guidelines for Improving the Software Process" (CMM V1.1), CMU SEI, Addison Wesley Publishers, Mass, 1994.

Artifacts. Products of a software effort that represent the various stages/phases of the effort. Typical artifacts include requirements specification, design specification, source code, executable code, test plan/cases/procedures/results.

Cultural. Human behavior and learned experience.

Derived Requirement. Those requirements (whatever their source - internal, supplier, team member, customer, etc.) that are generated apart from the primary requirements.

Formal Review. A formal meeting at which a product or document is presented to the user, customer, or other interested parties for comment and approval. It can also be a review of the management and technical activities and of the progress of the project.

Inspection. A static analysis technique that relies on visual examination of development products to detect errors, violations of development standards, and other problems. Error data is collected for later analysis and as a resource for future inspections. Types include code inspection, design inspection and test case inspection.

Iterative Build. The iterative build model might be a good choice for a project with unproven technology or a project with many user interactions because it provides a mechanism for fielding a working prototype sooner in the development process.

Lifecycle (Software). The period of time that begins when a software product is conceived and ends when the software is no longer available for use (is retired). The software lifecycle typically consists of a sequence of phases or stages such as concept, development, support, and retirement. There are various activities that can be conducted throughout these general phases depending upon the specific methods and techniques used. Examples of activities include planning, requirements definition, functional design, system design, programming, software integration and testing, installation and acceptance. Lifecycle methods and techniques such as the waterfall, prototyping, incremental, spiral, commercial-off-the-shelf (COTS), object-oriented, and so forth structure the phases or stages so that software activities are conducted in a particular manner and order in relation to each other.

Non-technical Issue. Agreements, conditions, and/or contractual terms that affect and determine the management activities of a software project.

Organization. As defined by SEI is a unit within a company or other entity within which many projects are managed as a whole. An operational definition of an "organization" is the scope of an appraisal or process improvement effort. Organizational analysis is necessary to define what the scope will be. Examples are a company, a division of a corporation, a government agency, and a branch of service. All projects within an organization share a common top-level manager and common policies.

Peer Review. A review of a work product, following defined procedures, by peers of the producers of the product for the purpose of identifying defects and improvements.

Political. Federal, state, and local laws and regulations, including copyright, patent, and trademark laws.

Quality Function Deployment (QFD). A quality management technique for translating customer quality requirements into technical requirements for each stage of product development and production. "The QFD Book: The Team Approach to Solving Problems and Satisfying Customers Through Quality Function Deployment" by L. Guinta and N. Praizler provides a full explanation.

Requirements Analysis. The process activities that derive specific requirements specifications for functions, behavior, performance, and interfaces from the context of an input information domain. Software-specific requirements analysis derives software-specific specifications.

Requirements Traceability Matrix. A representation of requirements specification information that provides a many-to-many map from/to each requirement to/from various system/software artifacts. The artifacts may include design, source, test, or other such materials. Minimally, a software requirements traceability matrix should indicate for each requirement, which test cases test that requirement and for each test case, which requirements are tested by that test case.

Social Factor. Different organizational communities such as business and information management or relationships of classifications, gender, age, power, etc. where communication difficulties can arise.

Software Manager: The software manager is responsible for ensuring the project is on schedule, is within budget, and has the requisite quality. The software manager is concerned with in-process measures that can provide early warning information. The software manager is responsible for the measurement planning information and for ensuring the measurement evidence is collected, analyzed, used in project decisions, and documented for lessons learned.

Software Engineer: The software engineer is responsible for developing and/or supporting the software product using systematic disciplined, quantifiable methods throughout the software lifecycle. Additionally, the software engineer may share with a quality engineer the responsibility for collecting process and product measurement data, analyzing the data to determine the impact on quality, and ensuring the measurement data and analysis results are properly communicated to the software manager. The software measurement data is used to determine whether process entry and exit criteria have been achieved.

Software Quality Engineer: The software quality engineer is responsible for ensuring quality is engineered into the engineering processes and resulting software products as acquired, developed, supported, or used by an organization or project. The software quality engineer may be part of a software organization or part of an independent quality assurance group, and should provide expertise in establishing a software measurement program. Measurement data provide in-process assurance and qualification evidence that requisite software quality has been attained.

Software Process Improvement Group: The software process improvement group is responsible for improving the software processes and resulting software products at an organization level. Improvement requires a measurement baseline and a continuous measurement program applied to organization projects to determine progress toward improvement goals. Within an SEI context, this is referred to as a Software Engineering Process Group (SEPG).

Software Customer: The software customer is any organization that receives a software product from a supplier for the purpose of qualification review, acceptance review, or operational use. The customer may be a Nuclear Weapons Complex organization or site, a Department of Energy organization or site, or an organization such as the Department of Defense. The customer is primarily interested in the cost of acquiring the software, schedule time to acquire the software, an operational software capability, software product defects that may be in the delivered software product, and on-going product support. The customer is responsible for the feedback of defects found during operational use.

Software Requirements Specification. The documented representation of the software requirements analysis activity.

Spiral Model. A software development process in which the constituent activities, typically requirements analysis, design, coding, integration, and testing are performed iteratively until the software product is complete. The spiral model provides a mechanism for supporting projects with many requirements that can be broken down into separate sub-components.

Structured Walkthrough. A review where an individual leads a group step-by-step or "walks them through" a specific product, document, segment of development, etc. For example, a designer or programmer may lead one or more members of a development team through a segment of design or code while the other members ask questions and make comments about technique, style, possible errors, development standards compliance, and other concerns. The walkthrough may or may not involve the customer and user.

Technical Issue. Those requirements that describe what the software must do and its operational constraints. Examples of technical requirements include functional, performance, interface, and quality requirements.

Validation. The process of evaluating a system or component during or at the end of the development activities to determine whether it satisfies specified customer requirements. The set of activities that ensure the correct product is built.

Verification. The process of evaluating a system or component to determine whether the artifacts of a given lifecycle satisfy the conditions imposed at the start of that phase. The set of activities that ensure a product is built correctly.

Waterfall Model. A software development process in which the constituent activities, typically requirements analysis, design, coding, integration, and testing are performed sequentially. The classic waterfall model can be used on a very simple project with a minimal requirements engineering task.

A.2 Acronyms and Abbreviations

ANSI American National Standards Institute
CMM Capability Maturity Model
COTS Commercial-Off-The-Shelf
DOE Department of Energy
IBM International Business Machine
IEEE Institute of Electrical and Electronics Engineers
NWC Nuclear Weapons Complex
QFD Quality Function Deployment
RM Requirements Management
SEI Software Engineering Institute
SEPG Software Engineering Process Group
SQAS Software Quality Assurance Subcommittee
V&V Verification and Validation

Appendix B: References and Bibliography

[BROOKS] Brooks, F. P., *The Mythical Man-Month: Essays on Software Engineering*, Addison-Wesley, Reading, MA, 1975.

[DEMING] Deming, W. E., *Out of the Crisis*, MIT Press, Cambridge, MA, 1986.

[DOESEM] *Software Engineering Methodology*, U.S. Department of Energy, March 1999.

[GARTNER] "Requirements Management: Taming Scope Scourge", The Gartner Group, April 15, 1997.

[GUINTA] L. Guinta and N. Praizler, *The QFD Book: The Team Approach to Solving Problems and Satisfying Customers Through Quality Function Deployment*. AMACOM Books, American Management Association, NY, NY, 1993.

[IEEE BP07738] *IEEE Software Requirements Engineering, Second Edition*, Edited by Richard H. Thayer and Merlin Dorfman, IEEE Computer Society, New York, NY. 1997

[IEEEstds] *IEEE Software Engineering Standards Collection: 1999 Edition*, IEEE Computer Society, New York, NY.

[IEEE610] IEEE Std-610.12-1990, The Institute of Electrical and Electronics Engineers Inc., *IEEE Standard Glossary of Software Engineering Terminology (ANSI)*, 1990.

[IEEE828] ANSI/IEEE Std 828-1998, The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Software Configuration Management Plans*, 1998.

[IEEE830] ANSI/IEEE Standard 830-1998, The Institute of Electrical and Electronics Engineers, Inc., *IEEE Recommended Practice for Software Requirements Specifications*, 1998.

[IEEE1233] IEEE-1233-1998, The Institute of Electrical and Electronics Engineers, Inc., *Guide for Developing System Requirements Specifications*, 1998.

[IEEE1233a] IEEE-1233a-1998, The Institute of Electrical and Electronics Engineers, Inc., *Guide for Developing System Requirements Specifications*, 1998.

[IEEE1028] IEEE-1028-1988, The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Software Reviews*, 1988.

[IEEE1042] ANSI/IEEE Std 1042-1987, The Institute of Electrical and Electronics Engineers, Inc., *IEEE Guide to Software Configuration Management,* 1987.

[IEEE1061] IEEE-061-1998, The Institute of Electrical and Electronics Engineers, Inc., *Standard for Software Quality Metrics Methodology*, 1998.

[ISO9000-3] ISO 9000-3:1991, "Quality Management and Quality Assurance Standards - Part 3: Guidelines for the Application of ISO 9001 to the Development, Supply and Maintenance of Software," International Standards Organization (ISO), 1991.

[KONTONYA] “Requirements Engineering With Viewpoints”, Gerald Kontonya and Ian Sommerville, *Software Requirements Engineering, Second Edition,* The Institute of Electrical and Electronics Engineers, Inc., 1996.

[SEICMM] Paulk, M.C., Curtis, B., Chrissis, M.B., and Weber, C.V., *Capability Maturity Model for Software, Version 1.1* (CMU/SEI-93-TR-024), Software Engineering Institute, Pittsburgh, PA, 1993.

[SQAS-GLOSS] SQAS-90-001, "NWC Glossary of Preferred Software Engineering Terminology," Software Quality Assurance Subcommittee, October 1990.

[SQAS-CM] SQAS-20.01.00, “Software Configuration Management (SCM), A Practical Guide”, Software Quality Assurance Subcommittee, February 2000.