

Managing Requirements across Multiple Related Products

Bhushan B. Gupta and Orhan Beckman, Ph.D.
Hewlett-Packard Company

Abstract

Managing requirements is a complex process. A requirements process typically involves elicitation and storage of requirements, assignment of requirements to a specific product, determination of dependencies on other products, creation of a product release plan, and monitoring of the progress toward meeting the requirements. An effective requirements management process must be able to manage this complexity as well as manage new requirements as they are received throughout development. It must also be able to accommodate the level of turmoil the iterative development practices place on it. The complexity of this process only increases when requirements have to be managed for multiple products that have inter-dependencies.

The requirements process described in this paper streamlines the gathering of meta-requirements (requirements for a requirements process), selection of a requirements tool, design of a requirements schema that facilitates efficient data retrieval, management of requirement dependencies between products, and the creation of development and release plans. The process focuses first on the characteristics of meta-requirements and requirements schema. The presentation covers the process of setting up this requirements management process, objectives, activities, and tools. In addition, the benefits realized and the challenges faced in practicing this method, are discussed.

Introduction

Requirements management entails “establishing and managing an agreement with the customer on a software project” (Paulk et al, 1995). Often, the customer is represented by the customer base already using the released products, users that fit the future product customer profile, customer surrogates such in marketing and support groups, and technical innovators in the development group. Requirements can be driven by current product enhancements and new feature requests from the customer base, new strategic marketing directions from the future product development, and the possibilities enabled through technological innovation. The customer is represented by these entities that constitute the primary stakeholders in the product. Product stakeholders are not limited to just these entities and may include those who develop the product, test the product and those who support and market the product once in the field. Since multiple groups depend upon requirements each should be consulted when developing and managing requirements. A good management process must consider these and other factors.

Requirement Management Objectives:

- **Adequate Customer Representation:** A sound requirements management process assures that the customer is adequately represented. All stakeholders participate in the elicitation, documentation, review, and prioritization of the requirements. For example, a support group contributes requirements that ensure the product is supportable, human factors engineers contribute requirements that ensure the product is usable and product marketing contributes requirements that ensure the product is marketable. The group responsible for testing will ensure each of the requirements is testable.
- **Well Defined Requirements:** Requirements should be unambiguous and properly defined with acceptance criteria. Good requirements are simple, measurable, attainable, reliable, and traceable (SMART). They can be written in English or using a requirement language but they should convey a consistent message. It is not a norm in the software industry to document the acceptance criteria while developing requirements. When followed, this

practice enhances the understanding of a requirement and forces each stakeholder to consider how the requirement will be met.

- **Well Rounded:** Requirements should focus not just on the product functionality but cover the entire spectrum of the quality monitors otherwise known as non-functional requirements. A conservative set of quality monitors includes functionality, performance, and reliability. If the software involves instant exposure to a large customer base (e.g. a web application), scalability and security may be added to the suite of quality monitors. Quality monitors provide a framework for requirements specification that help ensure the specification is complete. The monitors selected may vary from product to product and organization to organization.
- **Well Prioritized:** Not every requirement is created equal or represents a single group of stakeholders. Therefore, stakeholders participate in the prioritization of requirements to factor in the range of stakeholder needs. This prioritization aligns requirements with the marketing goals and strategies. In addition, the prioritization factors in customer impact.
- **Lead to Product Vintage:** Not all requirements can be delivered via a single product due to development schedule realities and resource constraints. Requirements therefore may be factored into, if not guide, product vintage planning. If prioritized based upon marketing needs and driven by customer surrogates, a product vintage plan will be realistic and one that is supported by the entire organization.
- **Adequate Tracking:** Since requirements are often shared among multiple products, (already in the field, being developed, and to be developed) they span an entire product vintage plan and therefore are traced. Traceability becomes increasingly important when an agile product development methodology is employed. In agile development a requirement may come in and go out of the scope within a single iteration.

Requirement Management Activities:

Requirements management activities include elicitation, specification and review, storage and tracking. Elicitation is the process of discovering what a product must do in order to meet customer needs. Conventional requirement elicitation techniques include internal analysis of the system under development to predict how users will interact with the system and interviewing users about their work today and problems they face in using existing tools and processes. But requirements derived from developers 'imagining' how the system will be used and even from current users describing how systems are used miss critical information about how systems are or will actually be used. To understand this delta and increase the validity of requirements it is recommended that the elicitation process include observation of how an existing system, or system under development, is used in addition to analytical or self-reporting measures. Understanding how customers work involves observation of them performing work tasks and rituals. To adequately predict usage patterns of a system under development, this involves prototyping the user interface and watching the user perform target tasks with the simulated system. For contextual validity both should be observed in the environment in which the current system is or target system will be used. In summary, effective requirements elicitation should utilize both analytical and empirical methods. An elicitation methodology that encompasses adequate customer representation and quality monitors has been described by Beckman et al [2]. The methodology follows a software product lifecycle and for each phase in the lifecycle it identifies the stakeholders that elicit requirements for each quality monitor.

Specification – The audience of requirements includes, developers, test engineers, learning products, and support groups. Requirements should be complete and follow a format that is usable by all the stakeholders.

Using a template to document requirements helps assure a level of uniformity and completeness to the requirements. A basic template covers the purpose of the requirement, its intended audience and a use case(s) that describes the behaviors the requirement will support. If a use case is adequately documented, it can facilitate the derivation of test cases [3]. Requirements

written jointly and reviewed by stakeholders, such as future product marketing and the development group promote a common understanding of the product and foster organizational cohesiveness. Once requirements are documented review them with stakeholders for accuracy and completeness.

Tracking – If a requirement is a simple enhancement to an existing project, it may be addressed in the following version. However, if it is a feature that requires substantial reengineering, it may not be addressed until a future version. This means that a requirement will be tracked for at least two versions of the product under development. There are situations when custom variations of an existing product require additional tracking. Other needs for tracking may arise from a product’s hardware or code-base dependency.

Requirement Lifecycle and Tracking Needs

A requirements management lifecycle guides organizational behavior and provides development support just like a defect lifecycle. An organization that strongly believes in defect prevention will adopt reviews and inspections of deliverables. Since the requirements are one of the most important deliverables for defect prevention, a review state in the lifecycle will help ensure significant requirements are reviewed.

A simple requirements lifecycle is illustrated below. The proceeding table explains the state transitions, pre-conditions, and post-conditions.

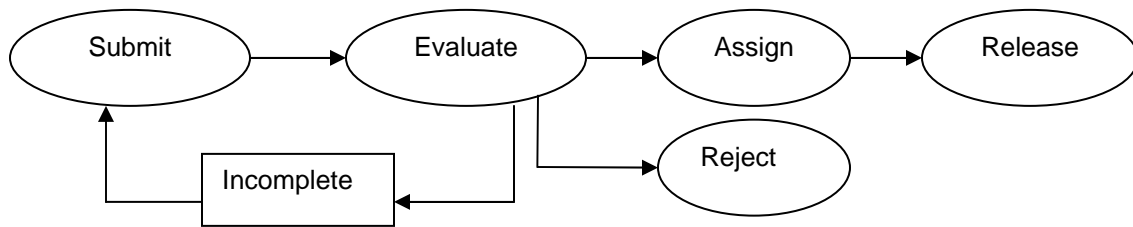


Fig. 1: A Basic Requirements Lifecycle

State From	Precondition	State To	Post-conditions
Submit	A user is ready to submit a requirements	Evaluate	The submitter has been communicated about the successful submit with a requirement ID.
Evaluate	A requirement has been submitted with the required fields.	Incomplete	An evaluator has determined that a requirement is not meaningful. The submitter has been communicated about the necessary changes
		Assign	Evaluator has determined that the requirement is meaningful and assigns it to all the relative products. If to be postponed include in the vintage.
		Reject	If not practical to implement.
Incomplete	The information has been corrected or missing information has been provided	Submit	
Assign	The requirement implementation	Released	The submitter has been informed

	has met the acceptance criteria		about the successful inclusion of the requirement in the product.
--	---------------------------------	--	---

Table 1: State Transition for Basic Requirements Lifecycle

Complexities in a requirements lifecycle may arise when activities such as review, prioritization and rejection are included. Requirement champions can ensure an adequate schema and tool are in place to support these characteristics.

Complexities when a requirement is to be met by multiple related products:

A customer requirement may be made up of several sub-requirements that are related to more than one product. Consider a web-based ordering requirement that applies to both order fulfillment and inventory management. If these two functionalities are supported by different subsystems that are on different schedules the following aspects of requirements management should be considered:

Requirements Submission:

It is desirable that a customer be able to submit requirements. However, this can lead to incomplete, ambiguous, and duplicate requirements. Karl Wiegers [4] defines a “product champion” as “a designated representative of a user class who supplies the user requirements for the group that he or she represents”. This person is most interested in delivering the right product to his or her user class. A product champion from each stakeholder group (future product marketing, development, and support) should form a team of product champions. The team of product champions submits requirements and ensures that they are clear, complete and concise.

Modularization and tracking:

- Break the requirement into subcomponents – This should be resolved by making sure that each component is self-sufficient and provides a unique value to the customer so that the subsystems can be released independently.
- Maintain a relationship between the main requirement and its subcomponents – This requires that the employed tool supports hierarchical dependencies such as parent and child relationships. The schema to support the tracking must include parent-child relationships at multiple levels if necessary.
- Track inter-dependencies between the components– The tool must support adequate tracking and provide any dependencies between two requirements..
- Make the state of each subcomponent visible through out the system – Design a set of queries that will provide the children and their states for the parent requirement.
- Set a parent requirement to “Released” only after all the subcomponents are in released state. A query that will list the release dates of subcomponents and generate the final release date for the parent is important.

Prioritization:

The product champion team understands customer needs and resources required to meet these needs. The future product marketing group champion understands what the customers needs, the support group champion knows the enhancements to be made to the existing product and the development group champion has the knowledge of the resources required to deliver the product The “team of champions” should be empowered to accept or reject and prioritize the requirements within a product and across the vintage.

Requirements Hierarchy:

Dependencies exist between requirements in a parent-child, or one-to-many, relationship. Dependencies also exist in situations where the nature of the relationship is many-to-many. Many-to-many relationship is a complex scenario and many tools may not support that.

Selecting a Requirement Management Tool:

Requirements management is complex in nature but can be eased by the use of a requirements management tool. Depending upon the details that need to be maintained, a tool can range from a simple spreadsheet to a comprehensive system that not only tracks requirements for related products, but also manages them throughout development lifecycles. A robust tool will support a requirements lifecycle with adequate messaging upon a state change.

When choosing a requirements management tool clearly identify the needs that the tool will address. Stakeholders should first establish the queries needed for tracking. Once the queries have been established, the necessary data for tracking can be derived from these queries. In an environment where a requirement is fulfilled by multiple products, we recommend that the following attributes should be tracked.

- Products affected by the requirement
- State of the requirement in each product
- Schedule impact on related products and delivery date.

A tool should store information needed to support queries. A useful query when working with related products is one that can find all the products related to a particular requirement. If a requirement is made of several sub-requirements that need to be tracked, it is helpful to establish parent/child relationship especially when the components are divided into multiple products.

There are several tools available in the market that not only track the requirements but also support development activities. A comprehensive list of these tools is available on the Internet [5]. While selecting a tool, stakeholders should clarify their needs and make a careful decision whether the team needs a requirement tool or a tool that also supports the product development lifecycle. The additional functionality can add administrative cost and complexity to the requirements management process.

Where do I start?

If requirements engineering is chaotic in your organization, the first step for improvement is to identify the elicitation techniques that will ensure quality requirements. The next stage is to appoint a product champion team that will ensure the requirements are SMART and adequately prioritized. Business planning normally includes creating a product vintage. When the number of requirements exceed the resources available for a product under development, it is necessary to assign requirements to a vintage.

Although a requirements management tool provides the ability to track progress and facilitates message communication, it also requires resources and may become a bottleneck. Many tools will help you meet your basic requirement management needs. Tools that provide additional capabilities often require dedicated resources. It is a good idea to assess your needs and invest in a tool that matches your needs.

Conclusion:

Requirements engineering is a critical aspect of product development. If stakeholders are not involved in requirement elicitation, prioritization and management, defect prevention activities will be less successful. This often results in poor product quality and a large amount of rework. A well thought out lifecycle with adequate communication can ensure robust requirements. A requirements engineering process that is carefully designed with adequate prioritization and defect prevention activities and is supported by a proper tool can lead to a sound product that meets customer needs and has a high level of quality.

References:

1. Paulk Mark, et al. 1995, The Capability Maturity Model, Guidelines for Improving the Software Process. Addison Wesley, Reading, MA
2. Requirements Elicitation for "Customer Delighting" Product Families, Beckman, Orhan. Ph.D., Gupta Bhushan, Sheffels Steve, 21st Pacific Northwest Software Quality Conference, October 2004, Portland, Oregon
3. Orhan Beckman and Bhushan B. Gupta, Developing Test Cases form Use Cases for Web Applications, Eighth International Conference on Practical Software Quality Techniques, March 2002, New Orleans, Louisiana
4. Wiegers, Karl E. Ph.D., Software Requirements, 2nd Edition, Microsoft Press, 2003
5. Ludwig Consulting Services Web Page
www.jiludwig.com/Requirements_Management_Tools.html