Requirements management

8.1 Requirements management is the process of documenting, analyzing, tracing, prioritizing and agreeing on requirements and then controlling change and communicating to relevant stakeholders. It is a continuous process throughout a project. A requirement is a capability to which a project outcome (product or service) should conform.

8.2 Overview

The purpose of requirements management is to ensure that an organization documents, verifies, and meets the needs and expectations of its customers and internal or external stakeholders. Requirements management begins with the analysis and elicitation of the objectives and constraints of the organization. Requirements management further includes supporting planning for requirements, integrating requirements and the organization for working with them (attributes for requirements), as well as relationships with other information delivering against requirements, and changes for these.

The traceability thus established is used in managing requirements to report back fulfilment of company and stakeholder interests in terms of compliance, completeness, coverage, and consistency. Traceabilities also support change management as part of requirements management in understanding the impacts of changes through requirements or other related elements (e.g., functional impacts through relations to functional architecture), and facilitating introducing these changes.[2]

Requirements management involves communication between the project team members and stakeholders, and adjustment to requirements changes throughout the course of the project.[3] To prevent one class of requirements from overriding another, constant communication among members of the development team is critical. For example, in software development for internal applications, the business has such strong needs that it may ignore user requirements, or believe that in creating use cases, the user requirements are being taken care of.

8.3 Traceability

Requirements traceability

Requirements traceability is a sub-discipline of requirements management within software development and systems engineering. Requirements

traceability is concerned with documenting the life of a requirement and providing bi-directional traceability between various associated requirements. It enables users to find the origin of each requirement and track every change that was made to this requirement. For this purpose, it may be necessary to document every change made to the requirement.

It has been argued that even the use of the requirement after the implemented features have been deployed and used should be traceable.[1]

Traceability as a general term is the "ability to chronologically interrelate the uniquely identifiable entities in a way that matters." The word *chronology* here reflects the use of the term in the context of tracking food from farm to shop, or drugs from factory to mouth. What matters in requirements management is not a *temporal* evolution so much as a *structural* evolution: a trace of where requirements are derived from, how they are satisfied, how they are tested, and what impact will result if they are changed.

Requirements come from different sources, like the business person ordering the product, the marketing manager and the actual user. These people all have different requirements on the product. Using requirements traceability, an implemented feature can be traced back to the person or group that wanted it during the requirements elicitation. This can be used during the development process to prioritize the requirement, determining how valuable the requirement is to a specific user. It can also be used after the deployment when user studies show that a feature is not used, to see why it was required in the first place.

Requirements Traceability is concerned with documenting the relationships between requirements and other development artifacts. Its purpose is to facilitate:

- the overall quality of the product(s) under development;
- the understanding of product under development and its artifact; and
- the ability to manage change.

Not only the requirements themselves should be traced but also the requirements relationship with all the artifacts associated with it, such as models, analysis results, test cases, test procedures, test results and documentation of all kinds. Even people and user groups associated with requirements should be traceable.

Definitions

A much cited definition of requirements traceability is the following:

Requirements traceability refers to the ability to describe and follow the life of a requirement, in both forwards and backwards direction (i.e. from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases.)[6]

While this definition emphasizes tracking the life of a requirement through all phases of development, it is not explicit in mentioning that traceability may document relationships between many kinds of development artifacts, such as requirements, specification statements, designs, tests, models and developed components. The next definition addresses this issue:

Requirements traceability refers to the ability to define, capture and follow the traces left by requirements on other elements of the software development environment and the trace left by those elements on requirements.[7]

The following definition emphasises the use of traceability to document the transformation of a requirement into successively concrete design and development artifacts:

In the requirements engineering field, traceability is about understanding how high-level requirements -- objectives, goals, aims, aspirations, expectations, needs -- are transformed into low-level requirements. It is therefore primarily concerned with the relationships between layers of information.[8]

The principal relationship referred to here may be characterised as "satisfaction": how is a requirement satisfied by other artifacts? Other relationships that can be traced are, for example, "verification": how is a requirement verified by test artifacts?

Tracing tools

There are several requirements management computer programs on the market for storing all requirements of all specifications of a technical system under development, which are arranged in a specification tree and linking each one to the "parent" requirement in the higher specification.

Evaluation functions allow for

- completeness checks i.e. do all system level requirements go down to equipment level (with or without modification)

- assessment of requirements deviations over all levels
- qualification status presentation

Tracing beyond the requirements

Requirements are realized into design artifacts, implementation, and are finally verified, the artifacts tied to the latter stages should be traced back to the requirements as well. This is typically done via a Requirements Traceability matrix.

Establishing traceability beyond requirements into design, implementation, and verification artifacts can become difficult.[9] When implementing software requirements for instance, the requirements may be in a requirements management tool, while the design artifacts may be in a tool such as MagicDraw, Matlab/Simulink, Rhapsody, or Microsoft Visio.

Furthermore, implementation artifacts will likely be in the form of source files, links to which can be established in various ways at various scopes. Verification artifacts such as those generated by internal tests or formal verification tools (i.e. The LDRA tool suite, Parasoft Concerto, SCADE)

Repository or tool stack integration can present a significant challenge to maintaining traceability in a dynamic system

Requirements traceability is concerned with documenting the life of a requirement. It should be possible to trace back to the origin of each requirement and every change made to the requirement should therefore be documented in order to achieve traceability. Even the use of the requirement after the implemented features have been deployed and used should be traceable.[4]

Requirements come from different sources, like the business person ordering the product, the marketing manager and the actual user. These people all have different requirements for the product. Using requirements traceability, an implemented feature can be traced back to the person or group that wanted it during the requirements elicitation. This can, for example, be used during the development process to prioritize the requirement, determining how valuable the requirement is to a specific user. It can also be used after the deployment when user studies show that a feature is not used, to see why it was required in the first place.

8.4 Requirements activities

At each stage in a development process, there are key requirements management activities and methods. To illustrate, consider a standard five-phase development process with Investigation, Feasibility, Design, Construction and Test, and Release stages.

Investigation

In Investigation, the first three classes of requirements are gathered from the users, from the business and from the development team. In each area, similar questions are asked; what are the goals, what are the constraints, what are the current tools or processes in place, and so on. Only when these requirements are well understood can functional requirements be developed.

In the common case, requirements cannot be fully defined at the beginning of the project. Some requirements will change, either because they simply weren't extracted, or because internal or external forces at work affect the project in mid-cycle.

The deliverable from the Investigation stage is a requirements document that has been approved by all members of the team. Later, in the thick of development, this document will be critical in preventing scope creep or unnecessary changes. As the system develops, each new feature opens a world of new possibilities, so the requirements specification anchors the team to the original vision and permits a controlled discussion of scope change.

While many organizations still use only documents to manage requirements, others manage their requirements baselines using software tools. These tools allow requirements to be managed in a database, and usually have functions to automate traceability (e.g., by allowing electronic links to be created between parent and child requirements, or between test cases and requirements), electronic baseline creation, version control, and change management. Usually such tools contain an export function that allows a specification document to be created by exporting the requirements data into a standard document application.[*citation needed*]

Feasibility

In the Feasibility stage, costs of the requirements are determined. For user requirements, the current cost of work is compared to the future projected costs once the new system is in place. Questions such as these are asked: "What are data entry errors costing us now?" Or "What is the cost of scrap due to operator error with the current interface?" Actually, the need for the

new tool is often recognized as these questions come to the attention of financial people in the organization.

Business costs would include, "What department has the budget for this?" "What is the expected rate of return on the new product in the marketplace?" "What's the internal rate of return in reducing costs of training and support if we make a new, easier-to-use system?"

Technical costs are related to software development costs and hardware costs. "Do we have the right people to create the tool?" "Do we need new equipment to support expanded software roles?" This last question is an important type. The team must inquire into whether the newest automated tools will add sufficient processing power to shift some of the burden from the user to the system in order to save people time.

The question also points out a fundamental point about requirements management. A human and a tool form a system, and this realization is especially important if the tool is a computer or a new application on a computer. The human mind excels in parallel processing and interpretation of trends with insufficient data. The CPU excels in serial processing and accurate mathematical computation. The overarching goal of the requirements management effort for a software project would thus be to make sure the work being automated gets assigned to the proper processor. For instance, "Don't make the human remember where she is in the interface. Make the interface report the human's location in the system at all times." Or "Don't make the human enter the same data in two screens. Make the system store the data and fill in the second screen as needed."

The deliverable from the Feasibility stage is the budget and schedule for the project.

Design

Assuming that costs are accurately determined and benefits to be gained are sufficiently large, the project can proceed to the Design stage. In Design, the main requirements management activity is comparing the results of the design against the requirements document to make sure that work is staying in scope.

Again, flexibility is paramount to success. Here's a classic story of scope change in mid-stream that actually worked well. Ford auto designers in the early '80s were expecting gasoline prices to hit $3.18 per gallon by the end of the decade. Midway through the design of the Ford Taurus, prices had centered to around $1.50 a

gallon. The design team decided they could build a larger, more comfortable, and more powerful car if the gas prices stayed low, so they redesigned the car. The Taurus launch set nationwide sales records when the new car came out, primarily because it was so roomy and comfortable to drive.

In most cases, however, departing from the original requirements to that degree does not work. So the requirements document becomes a critical tool that helps the team make decisions about design changes.[5]

Construction and test

In the construction and testing stage, the main activity of requirements management is to make sure that work and cost stay within schedule and budget, and that the emerging tool does in fact meet requirements. A main tool used in this stage is prototype construction and iterative testing. For a software application, the user interface can be created on paper and tested with potential users while the framework of the software is being built. Results of these tests are recorded in a user interface design guide and handed off to the design team when they are ready to develop the interface. This saves their time and makes their jobs much easier.

Release

Requirements management does not end with product release. From that point on, the data coming in about the application's acceptability is gathered and fed into the Investigation phase of the next generation or release. Thus the process begins again.

Requirements engineering

Requirements engineering (RE) refers to the process of formulating, documenting and maintaining software requirements[2] [3] and to the subfield of Software Engineering concerned with this process.

The first use of the term 'requirements engineering' was probably in 1979 in a TRW technical report [4] but did not come into general use until the 1990s with the publication of an IEEE Computer Society tutorial [5] and the establishment of a conference series on requirements engineering.

In the waterfall model,[6] requirements engineering is presented as the first phase of the development process. Later software development methods, including the

Rational Unified Process, Extreme Programming and Scrum assume that requirements engineering continues through the lifetime of a system.

Alan M. Davis maintains an extensive bibliography of requirements engineering.[7]

Requirements engineering activities

The activities involved in requirements engineering vary widely, depending on the type of system being developed and the specific practices of the organization(s) involved.[8] These may include:

1. Requirements inception or requirements elicitation -
2. Requirements identification - identifying new requirements
3. Requirements analysis and negotiation - checking requirements and resolving stakeholder conflicts
4. Requirements specification (Software Requirements Specification) - documenting the requirements in a requirements document
5. System modeling - deriving models of the system, often using a notation such as the Unified Modeling Language
6. Requirements validation - checking that the documented requirements and models are consistent and meet stakeholder needs
7. Requirements management - managing changes to the requirements as the system is developed and put into use

These are sometimes presented as chronological stages although, in practice, there is considerable interleaving of these activities