www.telelogic.com

WHAT IS REQUIREMENTS MANAGEMENT?

by Richard Stevens & James Martin

Comments from Ricka Harwell, Pradip Kar, Bob Smith, and Michele Bailey have been incorporated. This paper attempts to explore the nature of requirements management. The different tasks performed through requirements are illustrated, in order to show the "requirements for requirements".

Summary

Requirements management models the system, in order to improve the end product. A project that meets its requirements is by definition a success. Requirements management starts with the definition of requirements and continues through the project, culminating in the acceptance of the product against the requirements (Figure 1). By examining all the tasks that are performed with requirements, we can see the role of requirements management.

Requirements management could be defined as ensuring:

- we know what the customer wants (quality);
- the solution efficiently meets these requirements (conformance);

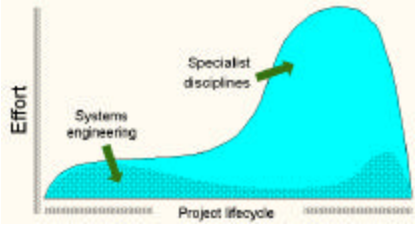

Figure 1: Systems engineers and specialists

Crosby's definition of quality is "conformance to requirements". Requirements Management is therefore the way in which we produce quality systems, starting with the definition of what is wanted from a system. This statement of the problem allows everyone to understand the basis of what they are doing. But requirements are alive and active throughout the lifecycle.

Quantitative figures are hard to obtain for the systems engineering role, as there is no firm definition of systems engineering. But experience with an organization whose prime role is requirements management suggests that systems engineering (of the customer, side role in general) represents 8-15% of the total project effort. Less than this results in loss of control for the customer. The higher figures become necessary for systems with a large amount of novelty.

Tasks of requirements management

- Define & communicate what is wanted
- Traceability to outside documents
- Apply requirements on the solution
- Optimize the product before commitment
- Drive the design & the implementation
- Manage change, problem reports, suggestions
- Manage the partitioning of work to specialists
- Testing & Validation of the finished product

- Control iterative developments
- Manage project milestones
- Manage interfaces with external systems

Requirements are a model, an early prototype of the system, which allows us to optimize the end product. The key feature here is that requirements are "live" and in use throughout the lifecycle.

Sometimes we treat requirements or traceability as if they were ends in themselves, without considering why effort is being expended on them. Requirements are used to model the system, so that we can improve the end products, making them do what they are meant to, eliminating mistakes in design and implementation. "Requirements for requirements" should be derived by seeing the uses of requirements throughout the lifecycle (Figure 2). Requirements engineering is meaningless outside the concept of a systems engineering lifecycle. It represents up-front work, for which the benefit does not appear until later. Unless there is a controlled systems engineering process in place, that benefit cannot be extracted from the requirements work. The following sections each cover one type of use of requirements within the systems engineering process.

Requirements to communicate what is wanted

The user requirements model defines what is wanted in non-technical terms. This model is used by everyone to see how their own work and role fits together. The term "users" applies to all those groups involved in operational environment of the system. Many projects fail although the solution is technically high quality. If a product does not do what the customer or the users want, then the build quality is irrelevant. User requirements define the results to be provided to users, and this can be done without a functional or behavioral definition of the system.

The key role of requirements is to show developers and users what is needed from a system. Provided the requirements are a language which everyone can understand, all people involved, including customers, can see the whole project and their role within that project. The first and most basic role of requirements is therefore communication.

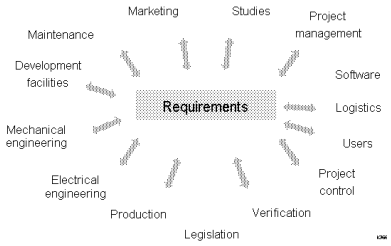

Figure 2: Requirements

Requirements applied on the solution

The systems requirements for a model of the system that does not commit us (too far) to implementation, yet provides an abstract view of the final product. Requirements on the solution represent a step forward from the definition of the problem. They form an abstract model of what is to be produced, expressed for example as a functionality. These requirements allow developers to see how their own work fits in with the system as a whole. This abstract model must allow different design options to be explored, and therefore must limit any commitment to a particular architecture.

Requirements for optimizing the system Any design must conform to the user requirements, and so the requirements model can help evaluate designs. Throughout the project life cycle, developers are faced with choices to cut the requirements, to delay parts of the implementation, and to choose between different options. These decisions should be made on a cost-benefit basis. A typical example occurs at the end of the systems requirements phase. At this point the requirements on the system are known, but no design has been chosen. The benefit to the user is "value of requirements" , and the cost is the cost of implementing the design. Provided if we have a requirements model, without commitment to design, it is possible to optimize the system design.

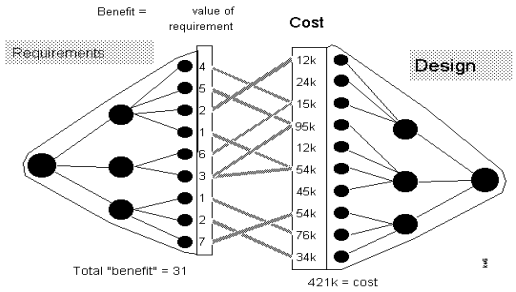

Figure 4: Cost-benefit evaluation through requirement

An alternative use for cost-benefit analysis is choosing between multiple options e.g. after a request for proposal (RFP). In this case, each reply will be evaluated to provide an overall benefit value against the proposed total cost.

Requirements to drive design and implementation

Requirements allow designers to imagine systems which will do what is wanted. Traceability between the design and the requirements lets us evaluate the design.

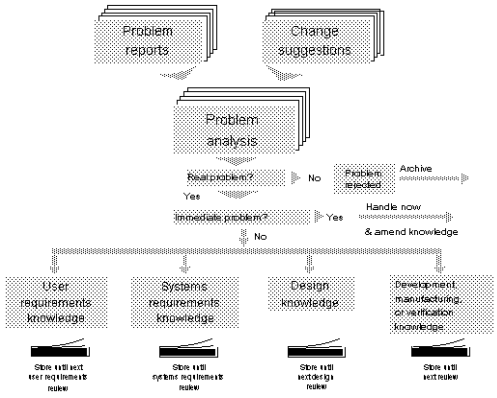

Figure 5: Change management in iterative systems

Requirements for managing change

Changes occur throughout the systems life cycle, and normally involve design or implementation problems that disturb the requirements. Unless we know the relationship between requirements and design, we have no rational basis for deciding on a change. A good set of requirements defines precisely what is wanted, but simultaneously leaves the maximum space for creative design.

Requirements for partitioning into sub-systems

A large systems development may involve 20-30 different groups of specialists, such as mechanical engineers, software engineers or project controllers. Each discipline has its own tools and languages. Large systems are difficult to manage. A major systems engineering role is to divide a single, large system into smaller sub-projects, which can then be developed in parallel without requiring too much systems level control. The system engineer bundles a set of requirements and passes it to the specialist implementors, which can then be developed in parallel without requiring too much systems level control. The system engineer bundles a set of requirements and passes it to the specialist implementors. Work can then be monitored against those requirements without needing to get involved in the detail and the languages of the specialists.

Each specialist group of engineers will use their own "languages", tools and notation to perform tasks such as stress analysis or software development. In contrast, the "language" and/or "notation" used by systems engineers must be simple for all developers to understand. This effectively restricts requirements to using natural language (or some sub-set) and simple diagrammatic techniques.

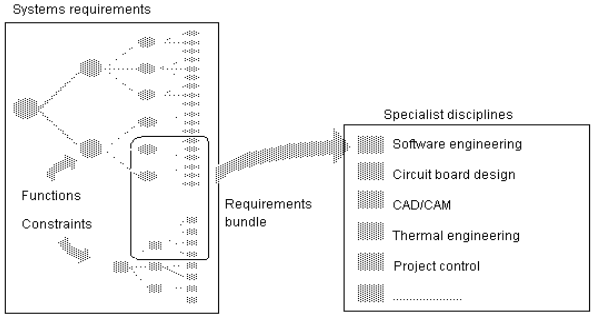

Figure 6: Partitioning from systems to specialist engineers

Requirements for testing the product

Requirements are the basis on which systems are finally accepted (or rejected). The final product is tested against the requirements.

The V-curve of verification shows that the different models used to define the product are used sequentially for verification.

Testing the system against requirements implies a relationship between product, test system and the requirements. Checking these relationships is a systems engineering task, although the detailed work may be sub-contracted.

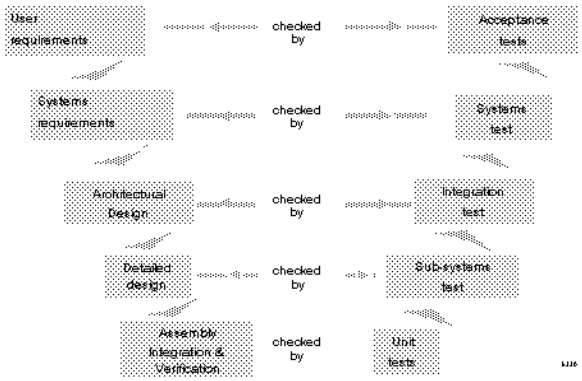

Figure 7: Requirements used for tests

Requirements to manage change of iterative systems

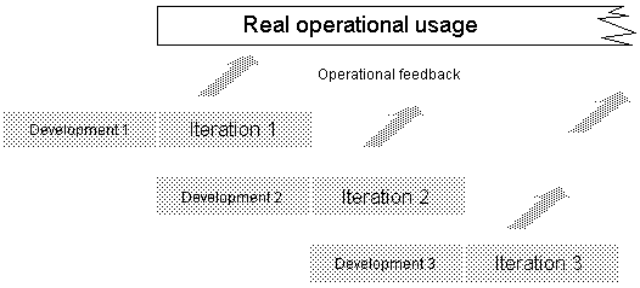

Figure 8: Feedback in iterative developments

Change management is a continual task for systems engineers (although change impact should be minimized). In iterative developments, the developers are involved in the working environment of a system. Suggestions for change and problem reports become the basis for driving future evolution. A problem is simply the inverse of a requirement.

In iterative developments, requirements form the basis of system evolution. Complaints and suggestions arising from real users of a product are particularly significant. In the single-shot life cycle, real operational users never have the chance to influence development, but iterative developments have parallel development and operational phases. Developers need to collect the information from users, prioritize and bundle them into development.

Requirements to assist project management

In some quarters, requirements management is seen as a technical issue, i.e concerned with the product, but not with project management issues such as schedule or resources. However, project management actually manages interdependent issues - schedule, resources, and quality (= conformance to requirements).

Imagine the following systems:

- "A concert that is high quality, built to budget, but is two days late"
- "A car that is high quality, built on time, but costs twice the market rate"
- "A system that meets the users and systems requirements, but has been developed using illegal labor"

These systems fail although they meet systems requirements. The failure involves lack of realism.
We want never to have to say "I've paid the money, but I haven't got what I wanted". To avoid this, requirements must be the basis of project management and financial control. For example, "finishing" a design does not constitute a milestone. But supplying a design that efficiently meets the requirements is a real milestone. Similarly, when a product meets the requirements under test or during operations, and the item is ready for input to the next stage a meaningful milestone has been achieved.

Every project milestone therefore alters the status of requirements, and the whole life cycle can therefore be monitored through requirements. Payment can be automatic against the achievement of real milestones.

The basic mechanism for creating milestones is to attach dates against configuration items. When the configuration item has been checked against its requirements, the milestone has been achieved. This requires traceability between the requirements and the configuration items to which they attach.

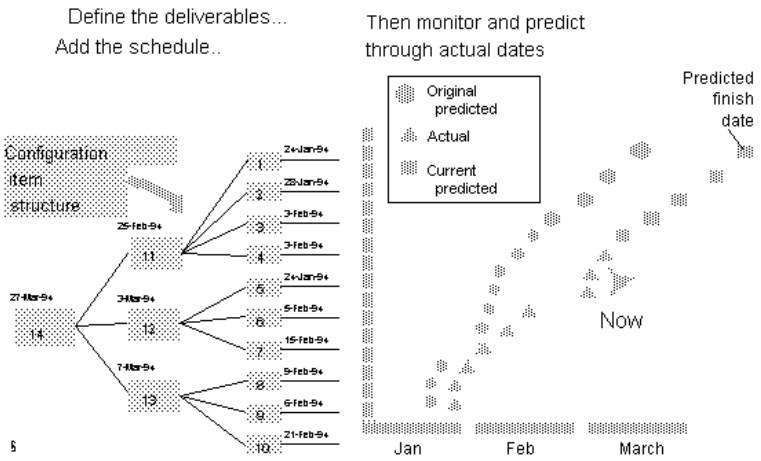

Figure 9: Requirements linked to configuration items

Summary of requirements for requirements

Several key factors emerge from examining how requirements are used. The first is that requirements are used throughout the life cycle - from beginning to end. That is not simply a "requirements phase" requirements have to be kept up-to-date throughout the project lifecycle. They are the whole basis of technical management.

The second feature is that requirements are inseparable from project management. If we accept milestones as the basis of monitoring progress, then we need to tie in requirements into the deliverables.

Requirements in abstract

Requirements models should exhibit a variety of characteristics - able to be produced early, cheap to make, easy to visualize and optimize, representative of the final solution. They must be complete - at some level of abstraction. Models should not commit us to a particular concept - they are for exploration and optimization. They should allows us to evaluate the final product. Unfortunately, these characteristics are mutually exclusive, and cannot be encapsulated in a single model. The early models we build with requirements are cheap to build, but are rather abstract and non-representative of the final product. Later the "prototypes" or qualification models that are built will be closer to the final product, but more costly and much less flexible.

Multiple requirements models

The numbers of requirements models that are built should be determined by the systems engineering standard used. Traditionally, there are two models for requirements - user and systems requirements - followed by a design. Unfortunately, we sometimes do not mentally separate the two fundamentally different concepts.

The user requirements models the problem or the results to be provided to the users. Users requirements are often organized as an operational scenario - a simple expression structured by time, stating the results to be provided to the users. Whoever produces the user requirements, they must be owned by the users.

The systems requirements model represents the solution in abstract, often through a functional or behavioral breakdown. Typically this model is organized by functionality or behavior.

Figure 10: Different models of the development

The design itself is a third (non-requirements) model. Clearly this is highly representative of the final solution. Many different designs can meet the requirements, and a change of solution may be possible without changing the requirements. In hardware systems, the manufactured product is, fourth model of the system.

Other system models

Building and maintaining multiple models involves effort, because they must all be kept aligned. If the model structure were the same as the final solution, this effort would be simplified. Some object-oriented approaches propose identical structure for systems requirements and design. For real systems, models must normally be structured differently because, for example:

- Non-functional requirements on one model causes changes in another;
- Re-use of existing components;
- Redundancy, spatial organization;
- Where different design architectures need exploration through a model which does not commit us to implementation.

Each of these factors will cause a design to be different to an early model. Where these effects are weak, it may be feasible to use a single model approach.

Note that a development will actually have many other models - for example, a configuration item structure models the deliverables, a work breakdown structure models the types of work. These are not "stand-alone" will actually have many other models - for example a configuration item structure models the deliverables, a work breakdown structure models the types of work. These are not "stand-alone" models - there are a variety of linkages between requirements and these models. These consist of, for example, linkages for:

Traceability - to prove that one system product is consistent with another (e.g., system requirements to user requirements).

Applicability - to link non-functional requirements to functional requirements.

Test linkages - to link the requirements to the test system (and hence to the product).

Development linkages - to link requirements to development models (e.g. WBS, OBS) or to link them to products (e.g., configuration item structure to design). This is needed to know, for example, which requirements apply to an individual deliverable.

What are requirements

The following section examines the characteristics of traditional requirements models to see how they fit against the requirements for requirements.

Characteristics of a requirements model

An individual requirement is meaningless in itself - it is a single element in a complete model. Some properties e.g., completeness or consistency are properties of the model as a whole. A requirements model must be a complete representation of what is known about the system at that point. The solution must meet some acceptable sub-set of the totality of requirements (perhaps the whole set). There is no point in meeting one single requirement. Failing to meet one key requirement may be catastrophic - a safety requirement for an airplane or climbing rope for example. A set of requirements forms a complete model, representing the system completely at some level of abstraction. The user requirements show the user the complete set of results to be obtained by using the system. The system requirement might show everything the system must do, plus all the constraints on that functionality.

Characteristics of requirements

A single requirement is an aspect of a system that must be true. The need may be abstract or concrete, it may apply to the product or to the way it is developed.

A requirement must be verifiable, i.e. it must be possible to imagine some kind of verification that checks the end product and gives us a binary answer "true" or "false". The objective of verification is proof of non-conformance as efficiently as possible. A requirement can never be proven to be met under test, although we might get close to proof. Other attributes include source, clarity, and priority.

Different kinds of linkage A substantial part of requirements work involves making links to other parts of the project. These should be a natural by-product of systems engineering. If we define a requirements model correctly, and check that the design meets the requirements, then we have traceability. One of the main roles of a good systems standard is to limit the linkages required. Traceability ensures that the various system models are consistent, for example that:

- Systems requirements meet the user requirements;
- Design meets the systems requirements.

Typically, links may be used to show:

- Applicability of non-functional requirements to functional requirements;
- Tests will check the appropriate design or requirements;
- Relationships between development information structures (e.g. between configuration item structure, schedule, and work breakdown structure);
- Cloning of information across the project;
- Each systems requirement will be tested;
- Rationale for critical requirements is documented.

Requirements tools must:

- Clear presentation of requirements in context
- Traceability, applicability, general linkages
- Ability to attach attributes to requirements & links
- Ability to link to descriptive information, e.g. studies
- Sorting & Selecting
- Text, graphics & publishing presentation
- Ability to incorporate users' knowledge of their own domain
- Automatic baselining, history, identifiers
- Ability to view at different levels of detail

Figure 11: Requirements for requirements tools (list)

Tools for requirements

Requirements management tools have to respond to the needs expressed in this paper, to help us organize requirements, create support tools to manipulate requirements-related information, and link to different areas of the project. Systems engineers have struggled to automate requirements management with simple tools, such as publishing systems with tags or relational databases.

Handling requirements involves managing complex sets of objects, related to almost every area of the project.

So what is requirements management?

This paper has covered the nature of requirements engineering, and requirements management is the task of ensuring that this work is done efficiently. Clearly the management of the requirements is a systems-level activity, covering both the product and the development (the way in which the product is developed).

Requirements are the basis of modeling the problem, then the product. These models are a form of prototype, that increases the chance of the final product satisfying users - i.e of being a quality product. Requirements management is the basis of quality.

Glossary

Validation: end-to-end verification i.e. conformance of the end product to the user requirements.
Verification: checking inputs of a phase to outputs, or that a product meets its specification.