

# REQUIREMENTS MANAGEMENT

Harold Halbleib

**We have all read the dire statistics where most IS-related projects are late, over budget, lacking in functionality, or never delivered. Requirements drive the development process. Effective requirements management helps control quality, cost, organization, and schedule, thus substantially improving the odds of a successful project. This issue's look at requirements planning begins with a thorough overview of the requirements management process.**

**R**EQUIREMENTS ARE THE AGREED-UPON facts about what an application or system must accomplish for its users. They are largely independent of the methods or notations used to analyze, design, implement, or test the target application. The process of managing requirements can determine project success and is especially important for larger systems, outsourced projects, and life-critical software. Although primarily focused on software development, much of this discussion also applies to other activities such as hardware development and systems engineering.

A typical project will have hundreds or even thousands of requirements. Identifying and specifying those requirements take time and effort, but the project payback can be enormous because they impact every aspect of a project, including design, implementation, testing, user documentation, and project management. A solid foundation can substantially reduce project risk and increase the efficiency of the entire development effort.

Historical data shows that defects occurring in the requirements identification phase are costly to correct. Requirement defects include missing, incomplete, nonessential, ambiguous, overlapping, and non-implemented requirements. A methodical approach to dealing with requirements can greatly reduce these deficiencies. Peers or customers can review

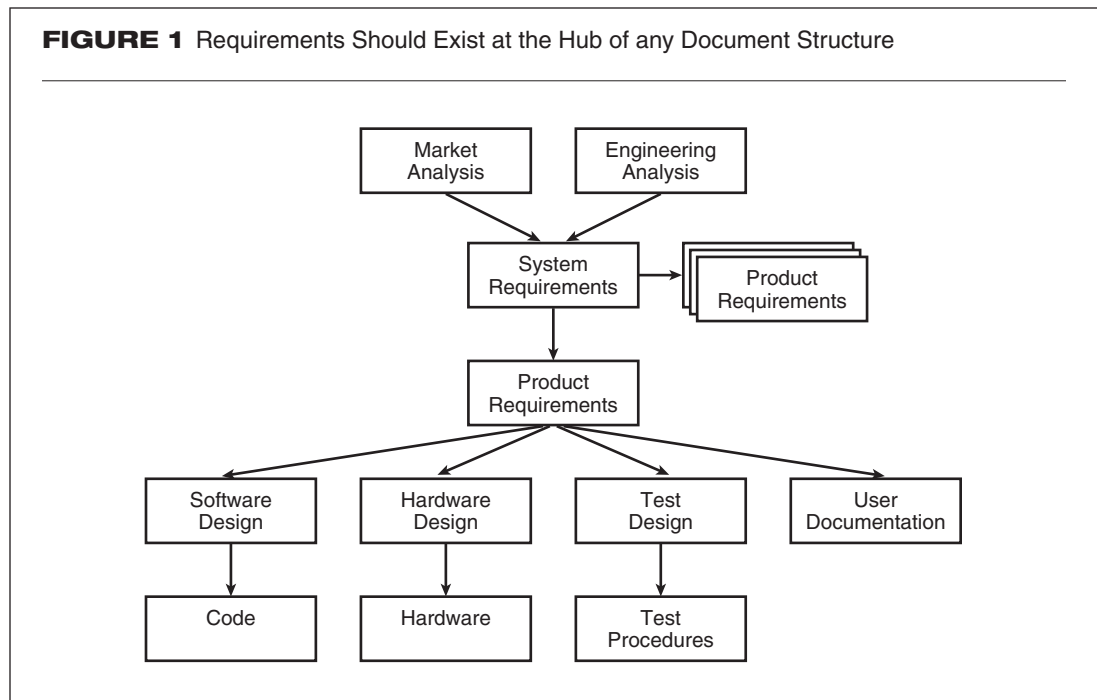
written requirements to expose gaps in the understanding of what the project must accomplish.

When developers spend time on design and implementation of nonessential or defective requirements, they are not just wasting time that delays the project. The added requirements often increase future maintenance costs, slow the final product execution, complicate the user experience, and make the project more rigid and susceptible to errors during future enhancements. An accurate requirements list keeps the development effort clean and focused.

Developers, testers, managers, and users need an organized approach to identify, specify, track, and control requirements. This article explores such an approach; it looks at the following components of requirements management:

- Definition, which identifies and groups requirements
- Process, by which requirements drive a project
- Structure, which defines a collection of information retained for each requirement entry
- Traceability, which allows navigation through project deliverables
- Automation, which can streamline the process of managing requirements

*HAROLD HALBLEIB has a degree in electrical engineering and ten years of experience in developing software for real-time process control systems. He has spent the past 15 years developing software engineering tools, training engineers, and working with hundreds of software companies around the world.*

**FIGURE 1** Requirements Should Exist at the Hub of any Document Structure**DEFINITION**

A requirement statement is an unambiguous, testable statement defining processing, information, performance, error handling, or capacity parameters about a condition or capability needed by a user to solve a problem or achieve an objective. System and product requirements are typically derived from market demands, quality, reliability, performance criteria, and safety considerations.

The first step in creating a list of requirements is to designate a definite boundary around the system to be considered. A simple picture or a few paragraphs of text can be a good starting point to ensure a clear, consistent mental picture. This boundary helps determine what belongs in the requirements list and what does not, because only externally visible aspects of the system must be included.

Each requirement is uniquely named and becomes one entry in a hierarchy list of requirements. Because a requirement entry can express a system requirement, product requirement, or UML-style use case for a wide variety of software systems, the granularity, style, and format of a requirement entry can vary significantly. It can be a simple, free format sentence or a more structured collection of data fields created with a user-defined template.

The names of each requirement entry can form a hierarchical structure. Usually, three levels of hierarchy are sufficient, wherein the first part indicates the major functional area in

which the requirement fits, the second part indicates the sub-functional area, and the last part makes the name unique. This grouping of related requirements makes them easier to locate and helps avoid incomplete or overlapping entries. When a large project is partitioned into sub-projects and assigned to different developers, the topmost level in the hierarchy can be used as a namespace. Developers can work independently in different namespaces. With automated tools, it is easy to collapse and expand the requirement hierarchy, import or export groups of requirements in a namespace, and generate various reports from the requirement information in a specific namespace.

For large, complex systems, it is often advantageous to develop separate system-level and product-level requirement documents. Each system requirement will likely reference multiple product requirements. The deliverables of development projects vary with each organization, but requirements should exist at the hub of any document structure (see Figure 1.)

For example, consider the design of a distributed process control system consisting of many individual products. A system requirement might specify timing constraints for communicating various types of messages from a process controller to an operator console connected on a data highway. The communication time depends on the processing time required

**D**esign, technology, and implementation issues are much more susceptible to change than true customer requirements.

by each device and across the data highway itself. Several individual product requirements, one for each device and one for the data highway, would be needed to satisfy this system requirement.

A requirement statement answers the question, “What must the system do?” or “What must the product do?,” but never, “How will the software do it?” Requirement statements are generally unaffected by the analysis, design, and implementation methods and notations used in the project.

Specifying how something will be accomplished in a requirement implies a design or, worse yet, an implementation. Different designs may satisfy the same requirements. A future redesign should have little impact on the underlying requirements. Design, technology, and implementation issues are much more susceptible to change than true customer requirements. A good requirements document provides a stable framework that maintains its integrity amidst those changes.

Occasionally, indisputable design constraints are placed on the project, such as conforming to an established architecture, industry standard or being compatible with an existing installed product base. In these cases, prior architectural or design constraints do become true requirements of the new development project.

A well-written requirement is testable. Avoid words such as “most” and “some” and adverbs and adjectives such as “quickly” and “robust” because these will make the requirement subject to interpretation and inherently non-testable. Be specific and watch for unstated assumptions. If a finite set of tests cannot prove that a system has passed or failed to satisfy the requirement, the requirement should be rewritten. Keep requirement statements short, focused, and nonredundant.

An important part of writing requirements is to develop a glossary for the project. Every organizational acronym, industry standard, piece of hardware, communication interface, protocol, and role that users can perform should become a defined name in the glossary. Always use glossary names in requirement statements to keep them short and nonredundant.

Having developed a list of requirements, you then want to review, refine, and rewrite this list to achieve quality definitions because they will drive team productivity during the design, implementation, and testing activities that follow. A formal requirements review

should ask the following questions. Are all requirements included, within the scope, and necessary for the software being developed? Is each requirement testable, feasible, independent, nonredundant, and traceable? Has any term been used that is not defined in the glossary with a single, unambiguous meaning? Is there any unnecessary information that can be stripped from a requirement to make it as terse and design independent as possible?

## PROCESS

Managing requirements is a process, not an event. That process starts at the conception of a project and continues until the developed system has been discontinued and is no longer supported. The structure of a requirement entry, where one field is its status, is discussed later.

The *Status* field indicates the state of progress of a requirement entry throughout its life cycle. The *Proposed state* identifies a newly suggested requirement. *Approved state* indicates that a requirement will be implemented in the current project. *Implemented state* means the requirement has been implemented and is ready for testing. *Validated state* means the requirement is completed. *Postponed state* indicates that the requirement will not be addressed by the current project.

An automated tool can track the progress of each requirement through each state of its life cycle. It should also allow one to add new states if the process requires it. For example, in a large system that includes system requirements and product requirements, one might want several validation states, such as *Unit Validation*, *Product Validation*, and *System Validation*. If outsourcing design and implementation, one might want an *Out Source Validation* and *In House Validation* state.

To manage expectations and achieve the project schedule, one will want to get most requirements nailed down very early in a development project. To reduce project risk, focus early prototyping efforts and detailed analysis on those aspects of a project with the largest unknowns and most significant impact on a project's success.

Requirements are identified and refined using numerous methods, including customer surveys, brainstorming sessions, marketing analysis, and engineering analysis activities. Dataflow diagrams and entity-relation diagrams are commonly used during this phase of the project. These diagrams provide a communication medium between the analyst, customer,

**R**equirements sometimes evolve during implementation and as understanding of the system and the user's needs change.

and management. They increase understanding of the problem domain and help to surface the essential requirements for the project.

Various methods and notations have been developed for analyzing systems and designing software. Structured analysis and design is popular in real-time, embedded systems. Desktop applications often use object-oriented analysis and design with notations such as UML. Data modeling has been used for decades to design database systems. Some diagramming techniques, such as DFDs and use case diagrams, are used during the process of discovering requirements while others, such as structure charts, class diagrams, and object diagrams, are used during design to satisfy specified requirements. Later we discuss how traceability ties requirement entries to analysis diagrams, design diagrams, code files, and test designs or procedures.

### STRUCTURE

In its simplest form, a requirements list could be one-line definitions, each having a unique entry name stored in a text file. Given the far-reaching impact of requirements and the different needs of users, developers, managers, and testers, one will likely want to gather a collection of information about each requirement entry. An automated tool makes this easy and should allow user-definable structure to that collection. Here is a standard structure that works well for most projects, along with the meaning and intent of each field.

The `Priority` field of a requirement can be set to `Low`, `Medium`, or `High` to indicate its urgency. This field is used in queries to show which requirements should get immediate attention. The `Status` field indicates the progress of a requirement entry throughout its life cycle, as discussed earlier. The `Author` and `Date` fields are usually defaulted by an automated tool at the time the requirement entry is created.

Use the `Assigned` field to indicate who has responsibility for implementing the requirement. One might want to add a `Tester` field if the organization has independent product testers. The project manager or team leader can assign test responsibility for each requirement entry.

The `Category` field lets one categorize each requirement entry. Category choices might include `Interface`, `Functional`, `Data`, `Configuration`, `Performance`, `Reliability`, `Compatibility`, and `Error`. One will probably need to adjust the

choices of this field in the user-defined template to better fit the type of projects undertaken.

The `Effort` field indicates in person-days an estimate of the work required for implementation. The organization needs to decide if that time estimate should include test effort, management overhead, technical documentation, etc. If grouping requirements into group and member entries, one may want to show the implementation time in each member entry and the remainder of the overhead time in the group entry.

The `Summary` field is a brief, single-line summarization of the requirement. Often when scanning through requirements or generating management reports, one will use this field rather than the full description. The `Description` field allows multiple lines of free format text where one can provide the explicit details of the requirement.

Requirements sometimes evolve during implementation and as understanding of the system and the user's needs change. The `Comments` field is a place for anyone to express suggestions, describe problems, or provide other feedback about a requirement. That information can later be considered when updating the `Description` field.

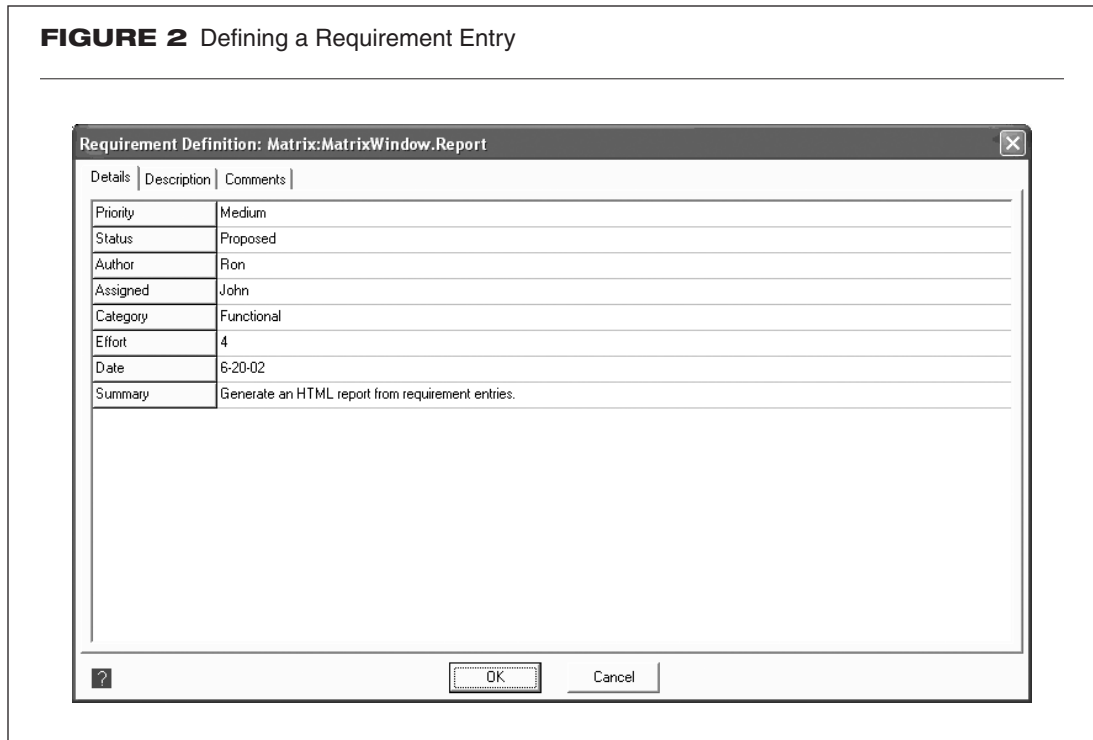
### TRACEABILITY

The `Parent` field of a requirement entry can link to earlier documents that provide supplemental information, marketing specifications, or engineering analysis. Documents later derived from the requirements are linked to the `Child` field. Requirements also link to each other to express a variety of relationships, such as inheritance, inclusion, and extension. Links assist in user navigation of project information and can be transferred into generated reports.

Each requirement can reference analysis models, design models, text specifications, code files, test files, HTML files, files created in other applications, and other requirement entries. For example, a system requirement will reference each related product requirement in its `Child` field, while each of those product requirements will reference the system requirement in its `Parent` field. Each requirement entry will probably target one or more graphic models from its `Child` field that illustrate its design.

Change control is an important aspect of requirements management. If requirements change, which documents are affected? Who needs to be notified? How significant is the ripple effect throughout existing design, code,

**FIGURE 2** Defining a Requirement Entry



and test procedures? Traceability improves risk assessment, project scheduling, and change control.

Traceability is a two-way street. If reading a requirement, the developer should be able to click to the related deliverables. Likewise, if looking at a design diagram of some type, one should be able to see all related requirements. Traceability also provides a navigation mechanism that allows one to select a requirement of interest and then navigate to the related design diagrams, code files, or test procedures in the project.

**AUTOMATION**

During the discussion of defining and managing requirements with a well-defined process, using a structured collection of data for each entry, and providing two-way traceability to other project deliverables, several automation opportunities were mentioned.

Gathering, refining, implementing, testing, and using requirements to manage a project is a collaborative process involving many people. Collaboration without a defined process can lead to chaos. However, following a process without automation is labor intensive.

Requirements management is an integral part of the development process that also includes system analysis and software design. Several tools are available to manage requirements, including Rational Software's Requisite

Pro, Telelogic's DOORs, and Excel Software's WinA&D product. WinA&D combines requirements management with structured analysis and design, object-oriented methods, and notations such as UML, data modeling for database systems, code generation, and reengineering. Screens from WinA&D shown in Figures 2 through 4 illustrate the automation of requirements management.

As shown in Figure 2, an automated tool can enforce a structure on requirement entries and make it easy to comply by just filling in the blanks. The fields in that structure should be configurable by the user, including allowable selections and data types of each field. Some fields like Author and Date can be automatically defaulted to reduce data entry time.

Requirement information should be viewable as standard or user-definable views and queries, as illustrated in the Requirement Matrix shown in Figure 3. Views determine what fields are shown in each column and their order. Queries define the selection criteria to determine which requirements are shown. By choosing a specific view and query, different users can precisely access the data they need.

For example, system analysts and software architects are primarily concerned with requirements in the Proposed or Approved state so they will use views and queries to reflect that. The project manager can view each requirement name and its priority, summary, status, estimated effort, and whom it is assigned

**FIGURE 3** Requirements Matrix Shows a Customized View of Requirement Information

Name	Priority	Status	Author	Assigned	Category	Effort	Date	Summary
Matrix:ManageQueries.Add	High	Validated	Ron	Tom	Configuration	3	6-13-02	Create a new view and present the
Matrix:ManageViews.Add	High	Validated	Ron	Tom	Configuration	1.5	6-13-02	Create a new view and present the
Rqts:TemplateManager.FieldsAndSec	High	Validated	Ron	Sue	Configuration	3	6-13-02	Partition a template into fields and se
Rqts:EntryVisibility	High	Validated	Ron	Tom	Functional	10	6-13-02	A tree structure can be toggled on c
Rqts:EditEntry	High	Validated	Ron	John	Functional	10	6-13-02	Provide a means of adding, viewing
Matrix:MatrixWindow.ResizeColumn	Medium	Validated	Ron	Tom	Functional	1	6-13-02	Allow user to resize matrix column.
Matrix:ManageQueries	High	Validated	Ron	Tom	Configuration	5	6-13-02	Allow user to configure named queri
Matrix:ManageViews	High	Validated	Ron	Tom	Configuration	5	6-13-02	Allow user to configure views that st
Matrix:ManageQueries.Edit	High	Validated	Ron	Tom	Configuration	2.5	6-13-02	Present editing dialog to create a us
Matrix:ManageQueries.Clone	Low	Validated	Ron	Tom	Configuration	1	6-13-02	Create a new view by cloning the cr
Matrix:ManageViews.Clone	Low	Validated	Ron	Tom	Configuration	1	6-13-02	Close a selected view and present t
Matrix:ManageQueries.Delete	High	Validated	Ron	Tom	Configuration	5	6-13-02	Delete a selected query.
Matrix:ManageViews.Delete	High	Validated	Ron	Tom	Configuration	1	6-13-02	Delete the selected view.
Matrix:MatrixWindow.Report	Medium	Proposed	Ron	John	Functional	4	6-20-02	Generate an HTML report from requ
Matrix:DataExchange	Medium	Approved	Ron	John	Compatibility	2	6-13-02	Provide a capability to transfer data
Matrix:ManageViews.Edit	High	Validated	Ron	Tom	Configuration	3	6-13-02	Presents an editing dialog to config
Matrix:MatrixWindow.Print	Low	Approved	Ron	John	Functional	2	6-14-02	Print requirement data from current \
Rqts:TemplateManager.EditingTempl	High	Validated	Ron	Sue	Configuration	2	6-13-02	Provide a dialog to edit and name up
Matrix:MatrixWindow.ReorderRows	Low	Proposed	Ron	Tom	Functional	1	6-13-02	Reorder rows when user clicks on c
Rqts:TemplateManager.FieldDefaults	Medium	Validated	Ron	Sue	Configuration	1	6-13-02	Replacement tags placed in a fl
Rqts:EntryVisibility.TreeStructure	High	Validated	Ron	Tom	Functional	5	6-13-02	Entry names can be displayed in a fl
Rqts:EditEntry.EntryName	Medium	Approved	Ron	John	Functional	2.5	6-13-02	The Entry Name dialog is used to re

to. That subset of information can be extracted to a scheduling program or edited to balance the workload of team members. Software designers and programmers can narrow their view to those requirement entries in a specific namespace reflecting the functional area they are working on or to those requirements specifically assigned to them. Programmers can move requirements from the Approved to Implemented state. Testers can view and focus their efforts on requirements assigned to them and, as completed, change the status to Validated.

In addition to displaying the precise information needed by each person, an automated tool makes that information exportable, importable, printable, and reportable using built-in customizable features. For example, a tool could include a built-in scriptable reporting engine with access to requirements, graphic analysis and design models, text specifications, dictionary information, code files, and test procedures. See [Figure 4](#) for a sample report.

Notice that the automation has not only enforced a disciplined, streamlined process where requirements flow through predefined states, but it has also reduced the effort of getting updated project status. At any instant, a project manager can generate a formatted project report that reflects the current status of

the project and highlights areas that need attention.

## SUMMARY

Requirements are an integral part of the development process connected through traceability with marketing specifications, analysis diagrams, design models, code files, test procedures, and virtually every other project deliverable. The structure of product requirements usually lends itself well to organizing the design, implementation, and test activities that follow. The mapping between system requirements and product requirements and between product requirements and design, implementation, and test procedures will help ensure complete coverage and eliminate overlap.

Each member of the development team uses a different subset of the requirement's information to do his or her job. An automated tool makes that information easily accessible through custom views and queries, user-defined entry dialogs and scriptable HTML reports. Given the leveraging aspect of the requirement's foundation, a good set of requirements is your best weapon in controlling project quality, cost, and schedule. Automated tools enforce a disciplined, streamlined process in that effort. ▲

**FIGURE 4** HTML Report Generated from Product Requirements

