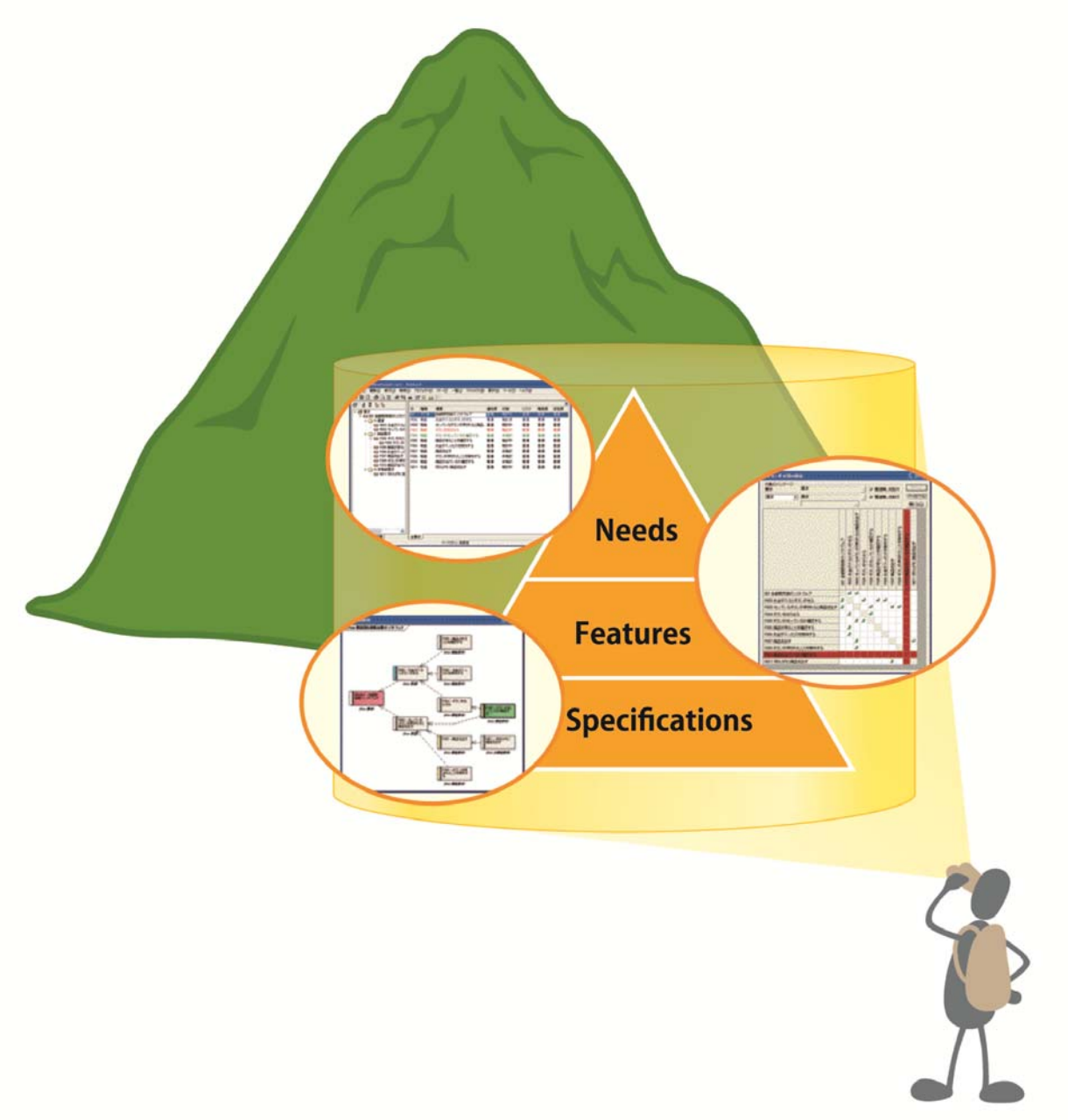
RaQuest
Requirements Management Tool

SPEI

Do you know? "7 Practices" for a Reliable Requirements Management

by Software Process Engineering Inc.

translated by Sparx Systems Japan Co., Ltd.



SPARX SYSTEMS

In this white paper, we focus on the "Requirements Management," which has a great impact on project's success and failure, and introduce its engineered concept and the practices with a tool. To quickly adapt the methods of requirements management onto projects, we introduce 7 practices by using one of the requirements management tools, "RaQuest."

Table of Contents

1. Introduction

Recent trends in requirements management

The term “Requirements Management” has increased in prominence in recent years, and the field’s relatively large body of literature has made it increasingly easy to gather relevant information. Whether the precepts of requirements management are being correctly applied to actual projects, however, remains unclear. Indications are that the concepts behind requirements management remain ambiguous, and are therefore difficult to apply in practice.

Requirements management is deeply related to themes that have a major impact on the success of a development project. Issues related to project scope are particularly closely related. Some problems, such as decisions concerning the extent to which requirements should be implemented before the next release, are quite difficult to solve without implementing methods from requirements management. Requirements management is thus no longer just a “nice thing to have” for development projects, but rather an absolute necessity.

Yet for most, while requirements management has now become a familiar term, its implementation has not yet taken root in actual projects. Given its importance to successful software development, this is an issue that should be addressed with some urgency.

Areas benefitting from application of requirements management tools

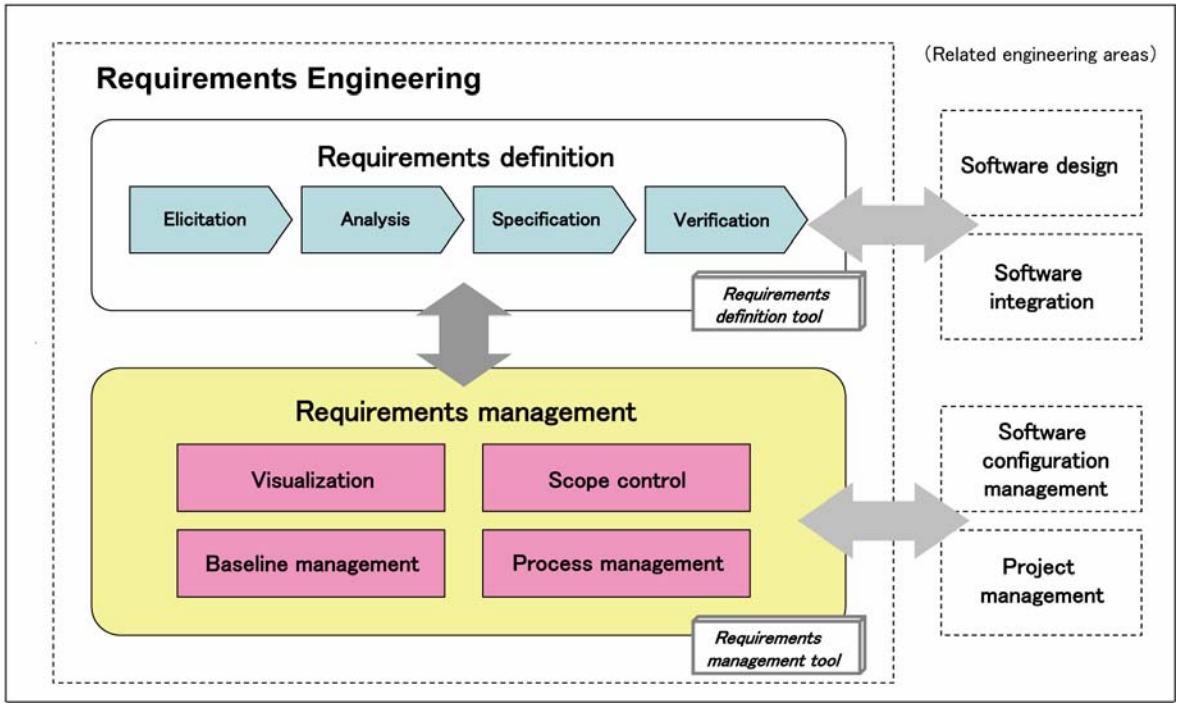
The application of tools in requirements management, the theme of this article, provides a hint as to how the issues described above can be addressed. The traditional definition of “tool” emphasizes the use of tools as a way of increasing efficiency in manual operations, but an additional aspect of tool use is better visualization of vague engineering concepts. Requirements management is an area where key concepts are particularly difficult to pin down, so here we look to tool use as an effective means for determining specific courses of action.

This article uses the functionality of one such tool, RaQuest, to introduce specific methods for cutting through the ambiguity and confusion surrounding requirements management, and putting it into actual practice.

2. Requirements management using RaQuest

The positioning of requirements management

Requirements management, along with requirements definition, falls under a wider domain known as *requirements engineering* (Figure 1). Requirements engineering, in turn, deals with all activities and techniques related to requirements, and is one of the areas that make up the practice of software engineering. The field of requirements engineering has established a number of best practices for determining requirement features and characteristics, and for dealing with them. Among those practices, some particularly important topics include visualization, scope control, baseline management, and process management. Further details of these topics can be read about in the technical literature, but in this article we will focus on showing how RaQuest features can be used to address each of them.



<fig. 1> Positioning of requirements management

Two tool types

As in other areas of software engineering, there are many tools available for supporting activities related to requirements engineering. Those related specifically to requirements can be roughly classified into two types: requirements definition tools and requirements management tools.

Requirements definition tools assist in specifying and documenting requirements, for example, UML modeling tools for describing Use Cases and drawing tools for creating data flow diagrams. Requirements management tools, on the other hand, generally have features that support tasks that are difficult to

perform manually with sufficient accuracy, such as requirements accumulation, visualizing, tracking, change management, and impact analysis.

RaQuest, a requirements management tool

RaQuest, discussed and used in this study, is one such requirements management tool. RaQuest incorporates many of the concepts and best practices of requirements engineering, making it an optimal tool for users having a basic understanding of the methods of requirements engineering and the necessity of requirements management, but are unsure of the specifics of how to implement them.

"RaQuest" is a requirements management tool that Sparx Systems Japan develops and distributes.
RaQuest version 3.3 is used in this white paper.

<Product information> http://www.raquest.com/

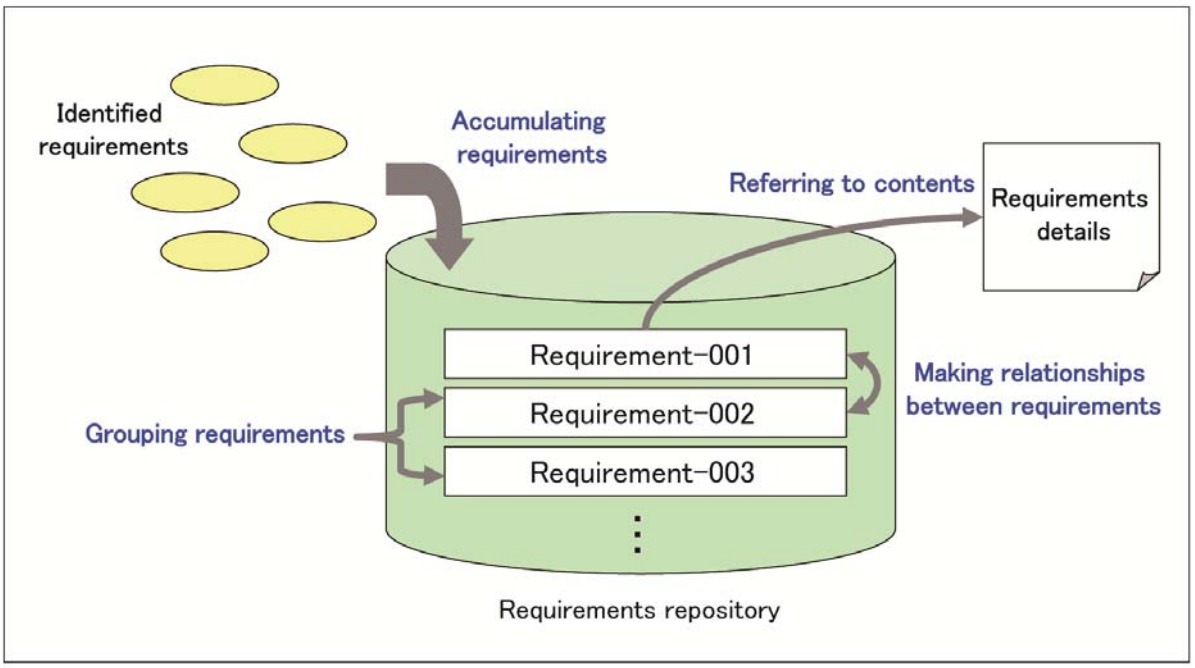
In what follows, the primary functionality of RaQuest is used to show the details of how it is used to address important topics in requirements management.

7 practices of RaQuest in this whitepaper	Topics of corresponding requirements management
【Practice 1】Accumulate requirements in a repository	Requirements visualization Process management
【Practice 2】Classify requirements by package	Requirements scope control
【Practice 3】Verify requirements through traceability	Requirements visualization
【Practice 4】Use requirement attributes to visualize implicit knowledge	Requirements visualization
【Practice 5】Control projects according to requirement status	Process management Requirements visualization
【Practice 6】Use baselines to represent project scope	Requirements baseline management Requirements scope control
【Practice 7】Maintain baselines through requirement change management	Process management Requirements baseline management

3. Seven practices for requirements management

<Practice 1> Accumulate requirements in a repository

Requirements for software development are frequently collected during discussions with users, and recorded on paper, on whiteboards, or in spreadsheets. When this is done strictly for follow-up verification or a quick explanation, such temporary records of identified requirements are likely sufficient. If the goal is to make use of the identified requirements throughout the entire software development lifecycle, however, all requirements need to be gathered into a single location and maintained in a format that allows them to be easily viewed. In other words, requirements organization needs to occur in a manner similar to a repository, not a file server. Storage in a repository allows for individual identification, proper referencing, and editing. More advanced functions, such as linking related requirements into groups, make repositories an invaluable environment for handling requirement sets.

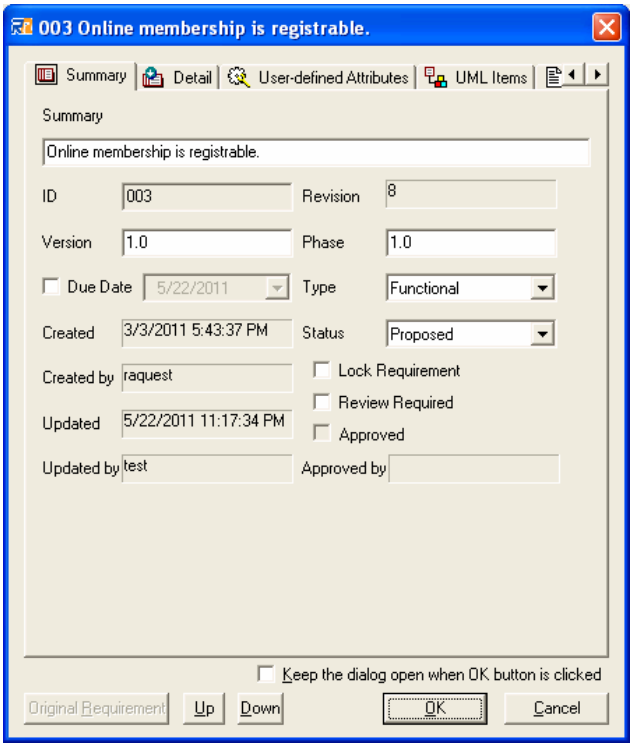


<fig. 2> Accumulate requirements in a repository

When requirements are entered into RaQuest, they are stored in a repository as manageable data elements. Unlike information written on paper or typed into a spreadsheet, requirements stored in a repository become basic data that can be utilized for a variety of goals. Having requirements stored in a repository is one of the most basic conditions for successful requirements management.

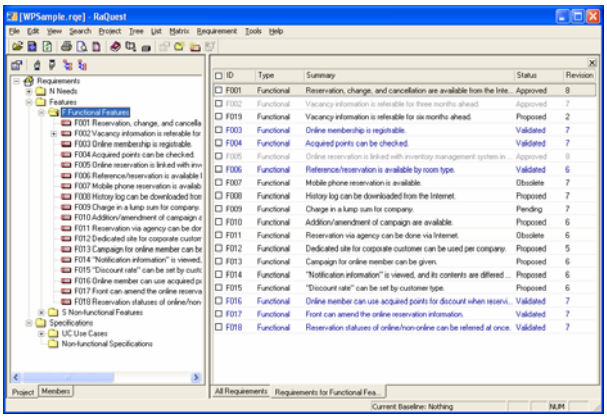
Creating requirements in RaQuest provides a number of benefits. For example, each requirement is automatically assigned a unique ID, a necessary but tedious part of requirements management. Major input supports include the following <screen 1>:

- Automatic ID assignment for requirements and revisions
- Versioning and phase settings
- Automatic recording of creation and change
- Automatic update log maintenance

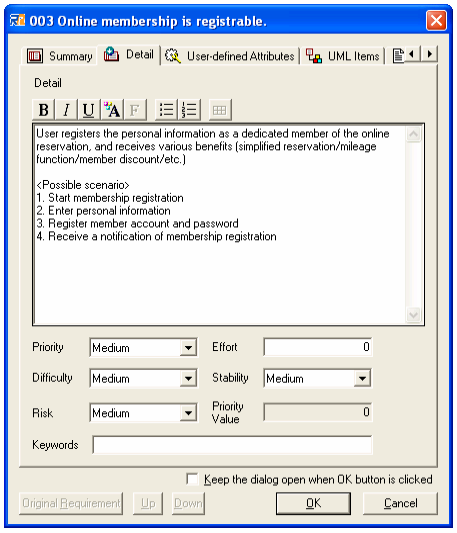


<screen 1> Creating a new requirement

Because RaQuest stores requirements in a repository, they can be examined in a various ways. For example, they can be displayed in tabular form like in a spreadsheet, or with extensive details shown like in word processing software. Each view is just a rearrangement of items stored in the repository, so requirements inconsistency and other features are always maintained. The result is that inconsistencies resulting from partial editing of similar data replicated across a number of formats are avoided.



<screen 2> List of requirements



<screen 3> Details of individual requirement

< Requirements management practice 1> Accumulate requirements in a repository

- All requirements must be stored within a data repository.
- Requirements viewing must be performed using items stored in the repository.

<Practice 2> Classify requirements by package

When the number of requirements becomes large, finding a mechanism to manage them becomes an important issue. From a requirements engineering perspective, requirements are generally classified according to level of abstractness and type.

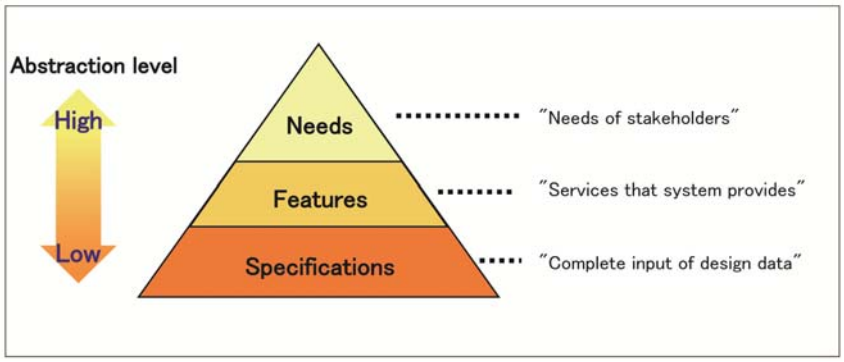
- Level of abstraction

A requirement's level of abstraction depends on factors such as the detail level, the source of the requirement, and the timing of its recognition. When managing requirements, identifying a requirement's level of abstraction allows for clear comprehension of the level of a requirement under consideration, and allows for smoother progress in defining requirements. The following describes three types of requirement abstraction:

Needs *refer to stakeholder needs. This is the highest level of abstraction, and the basis from which* Features *and* Specifications *(below) are formed. These reflect comments and opinions obtained directly from stakeholders, and must reflect the true needs of users of the software. When user needs are ignored, the result can be a failure to identify the requirements and specifications necessary to create software that accomplishes the desired tasks, and therefore becomes unusable.*

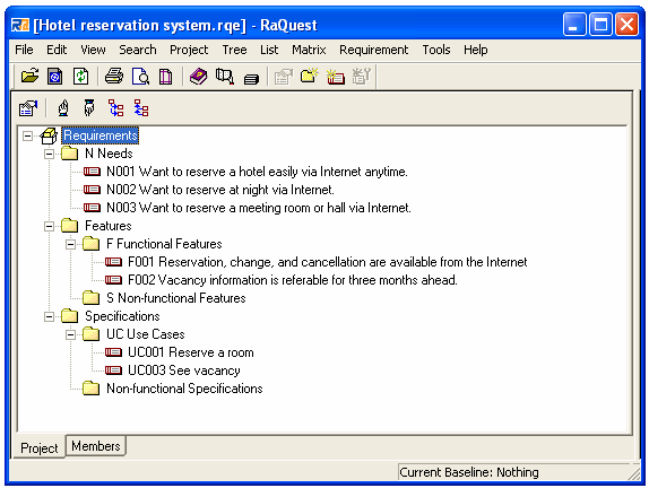
Features *refer to the capabilities and services provided to the user, or the characteristics that the software should have. Each feature must meet at least one of the specified needs, such that statements like "Feature* X *is implemented to fulfill Need* Y*" are possible.*

Specifications *refer to requirements defined and documented at a level of specificity that allows them to be treated as an input for system design. This includes functional specifications, interface specifications including screen elements and form layout, and data specifications. Specifications will eventually be compiled into a formal document known as the requirements specification. This document gives precise details related to the features and capabilities the system must provide, as well as constraints that must be adhered to. It provides the basis upon which further development activities such as design, implementation, and testing are performed.*



<fig 3> Requirements abstraction level

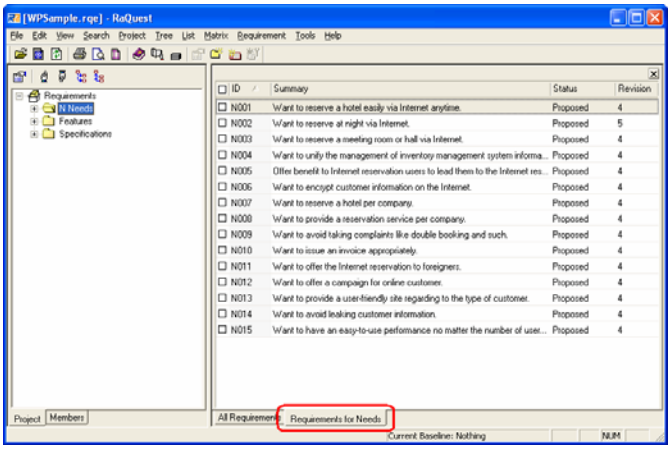
RaQuest represents requirement abstraction through its *Packages* functions. Packages are a way to bundle and store separate requirements into single units, similar to how UML packages are used. Packages can be freely created, so preparing needs, features, and specifications packages and appropriately sorting requirements among them is one way of managing requirements abstraction as described above. <screen 4, 5>



<screen 4> Package tree by the abstraction levels

Sorting requirements into Packages also allows other RaQuest functions to be applied to multiple requirements as a single unit. The following are some package-related functions that RaQuest provides:

- Listing requirements by package
- Changing package display items
- Issuing requirement ID by package
- Exporting requirements by package



<screen 5> Displaying the list by packages

- Requirements classification

The second requirements abstraction is the classification type. Requirements are broadly sorted into two types, *functional* and *non-functional*. Functional requirements are observable behaviors that result from the system itself, while non-functional requirements result from fixed attributes and qualities that the system must have. Use of requirement types makes it easier to understand what category of requirement needs to be collected, and helps to prevent omission from collections.

Functional requirements	"Functionality"	Actions that system must have to provide beneficial functionalities for users
Non-functional requirements	"Usability"	Requirements for integrity of user interface, learnability, and necessity of of help/support documentations
	"Performance"	Requirements for response time, throughput, capacity, and resource efficiency
	"Reliability"	Requirements for applicability, availability, data accuracy, system security, and data protection
	"Security"	Requirements for confidentiality, storage stability, and security availability
	"Maintainability"	Requirements for changeability, portability, reusability, and localization possibility
	"Operability"	Requirements for operability, and monitoring availability
	"System Constraint"	Constraints for the technology limitation, and compliance to laws/regulations

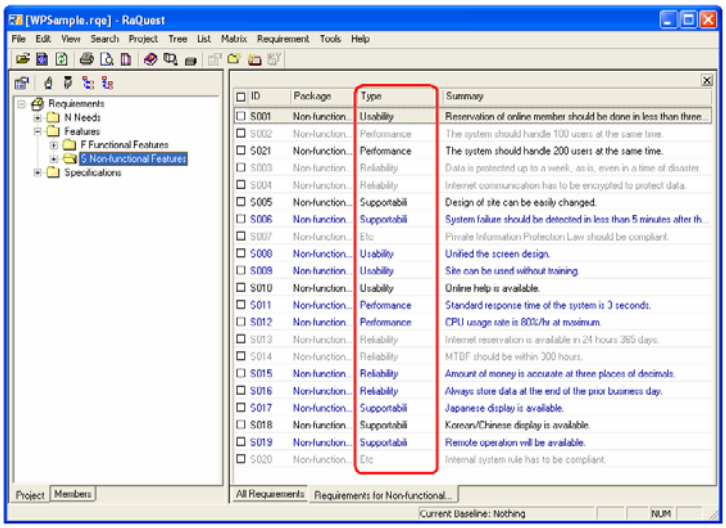
<fig. 4> Requirements classification

Figure 4 shows one example of requirement type categorization, but many other classification approaches are also used. Below we give FURPS+ (Robert Grady, 1992) and ISO/IEC 9126 (JIS X 0129-1) “Software engineering—Product quality” as examples.

- Classification of requirement on FURPS+ (Robert Grady, 1992.)
F: Function, U: Usability, R: Reliability, P: Performance, S: Security, +: Other

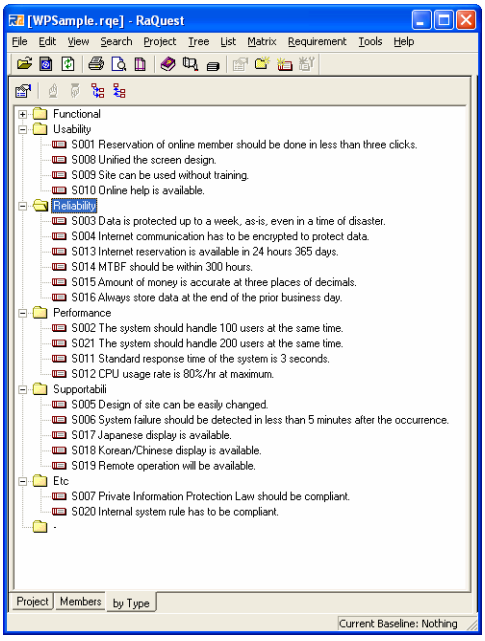
- Requirement classification on ISO/IEC 9126 Software engineering - Product quality.
Functionality, Reliability, Usability, Efficiency, Maintainability, Portability

RaQuest supports requirement typing using its *Type* function, which allows assigning types to individual requirements to sort them into categories according to the classification model used. RaQuest comes with the FURPS+ model by default, allowing you to begin categorization immediately, without creating your own model. <screen 6>



<screen 6> List of non-functional requirements after specifying the requirements type

RaQuest also has a custom tree function, allowing display of requirements trees in various ways. Using custom trees is a good way to change requirement representation without changes to the repository structure. For example, when you want to focus on non-functional requirements, you can use this function to display the requirements tree by type. <screen 7>



<screen 7> Package structure of requirements by the custom tree

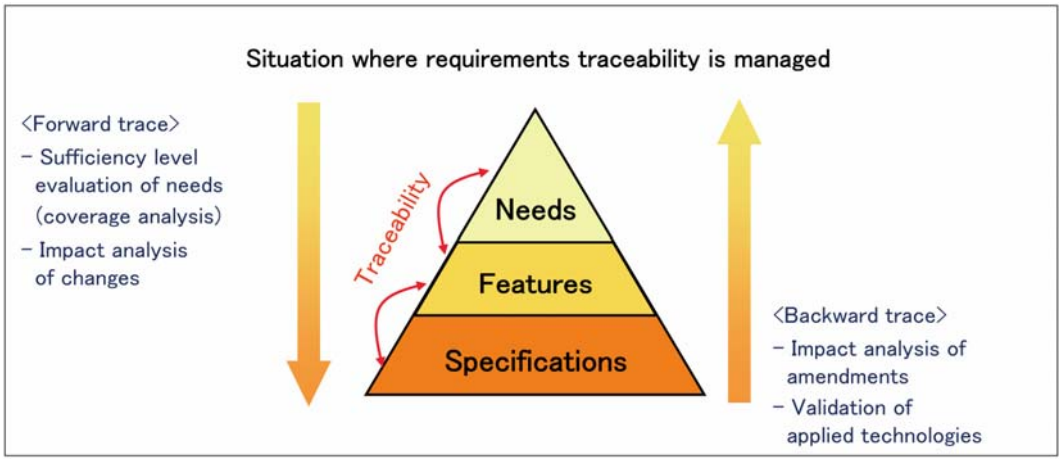
< Requirements management practice 2> Sort requirements into packages

- Make requirements distinguishable by level of abstractness
- Sort requirements into functional and non-functional types

<Practice 3> Verify requirements through traceability

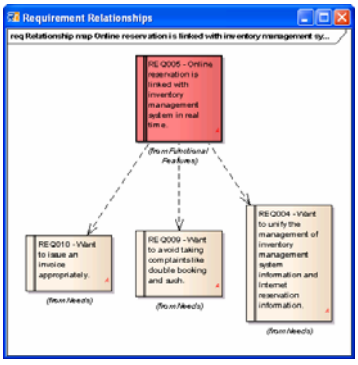
Another commonly implemented practice in requirements management is requirement verification through the use of traceability. Simply put, this refers to making it possible to follow requirements to other requirements or to operational areas, such as designs and implementations. When a requirement is to be changed, traceability allows easier impact analysis to investigate the effects on other requirements, as well as coverage analysis to verify that needs are being fulfilled.

As the number of requirements increases the number of relationships between them increases exponentially, making it nearly impossible to manually implement traceability. Traceability is important not only when requirements are being defined, but throughout the entire software development lifecycle. Because it is difficult to maintain a high level of quality when manually managing traceability, the use of tools that provide continual support is vital.



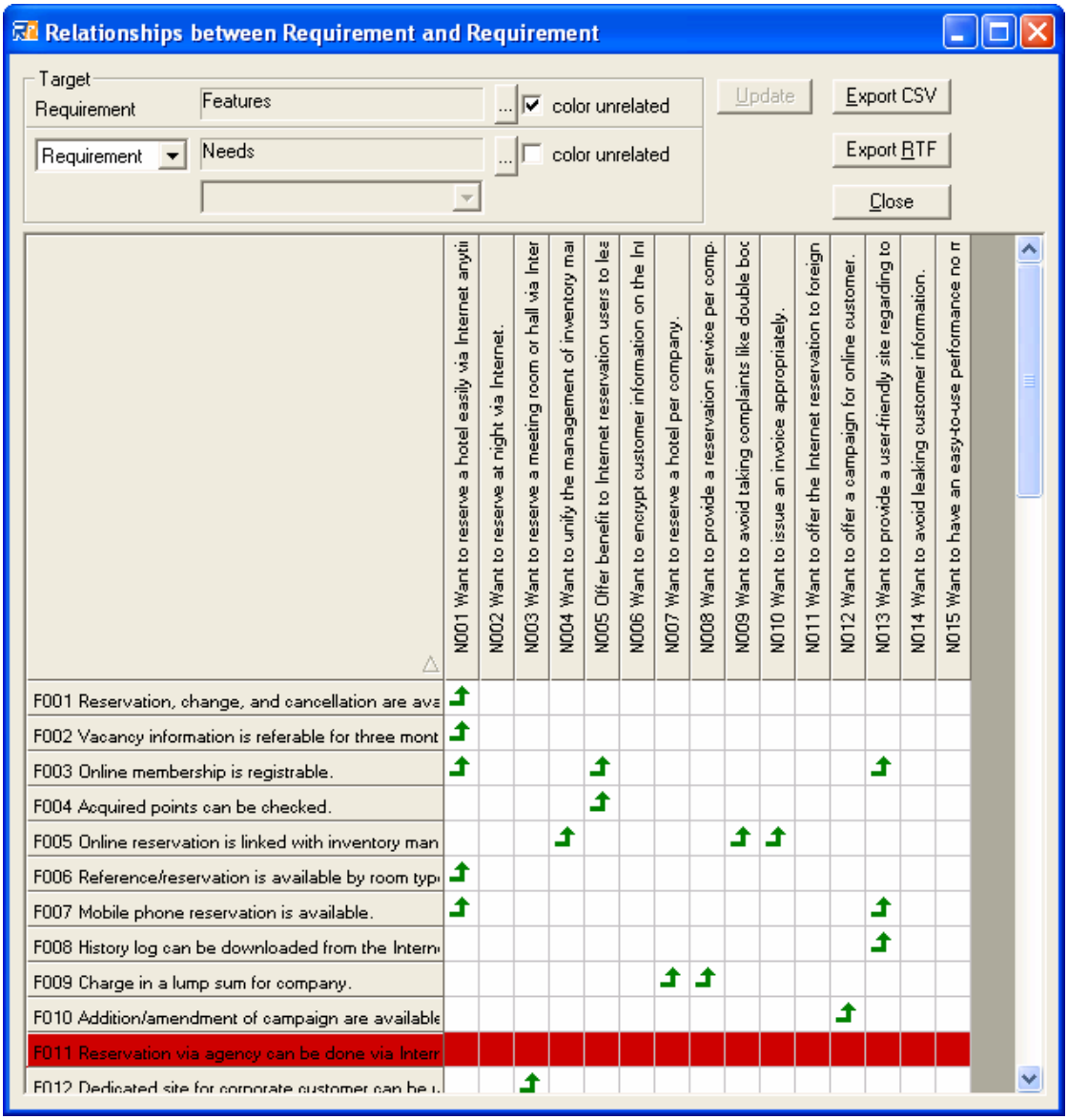
<fig. 5> Requirements traceability

RaQuest has a number of functions that support requirement traceability. Impact analysis and coverage analysis are supported from initial requirement creation through the addition of relations between requirements. Establishing these relations makes it possible to reference requirement relations through a variety of views. When performing impact analysis, there are many cases where traceability needs to be referenced for each individual requirement. RaQuest's relation map display function allows this to be done instantly. <screen 8>



<screen 8> Show a relationships map for the impact analysis

When performing coverage analysis to verify completeness between requirements at different abstraction levels, RaQuest's traceability matrix function allows the relations between multiple requirements to be viewed at a glance. For example, to verify that all identified needs have associated requirements, displaying the traceability matrix allows a visual inspection of requirement status. The traceability matrix also has an update function, allowing addition or deletion of requirements during verification. <screen 9>



<screen 9> Coverage analysis by the traceability matrix

< Requirements management practice 3> Verify requirements using traceability

- Assign relations between requirements when they are created
- Use a traceability matrix to verify complete coverage of requirements

<Practice 4> Use requirement attributes to visualize implicit knowledge

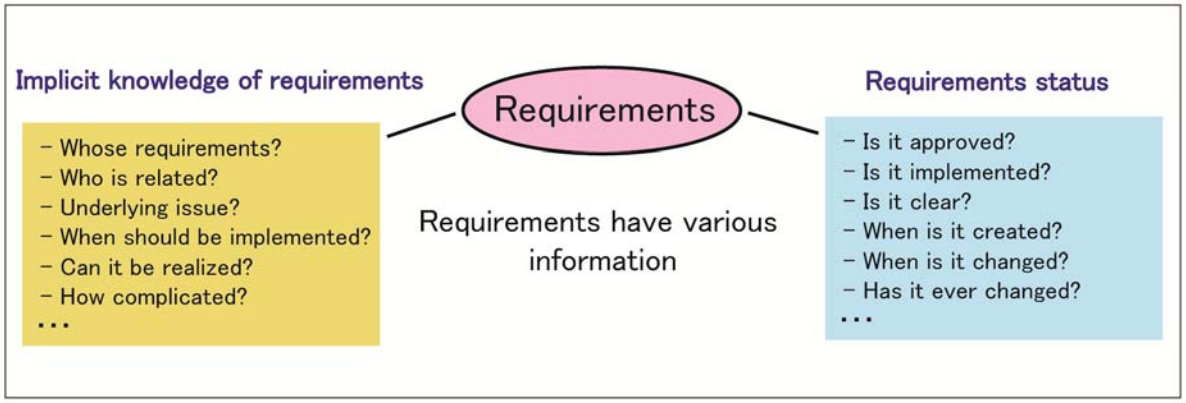
Stakeholder recognition of requirements differs according their point of view, but such differences are often held as implicit knowledge. Such differences can be made evident by assigning requirement attributes to each requirement, thus bringing forth perception gaps between stakeholders, and their differing levels of risk. Assigning requirement attributes helps to visualize tradeoffs between requirements and allows for objective evaluations of priority. Frequently used attributes include the following:

Priority *is the most general requirement attribute, and is used in many projects. This attribute provides objective criteria for selecting requirement implementation, serving as an indicator of requirements validation.*

Difficulty *is an evaluation of the risk associated with the realization and implementation of requirements, and represents an evaluation of requirements from a technological standpoint.*

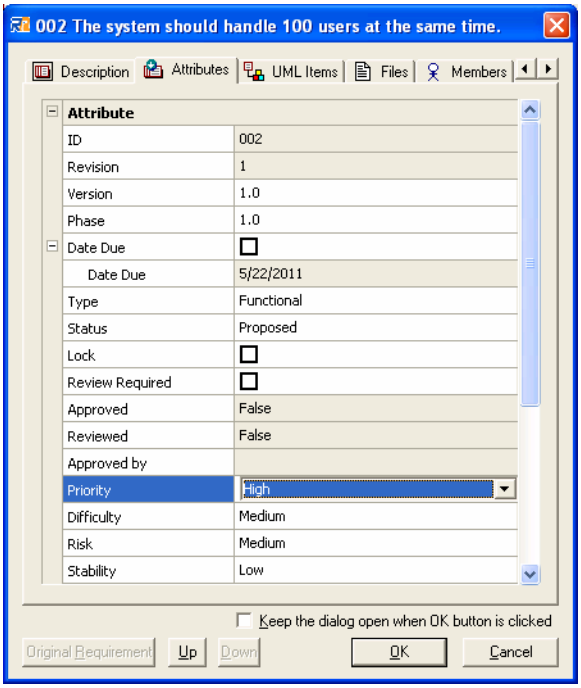
Cost *is used to give a rough idea of scale within the system, representing scale as measured through the use of some index.*

Requirement stability *is used as an identifier of the underlying potential risk based on the likelihood that the requirement will change, as well as the potential necessity of addressing such changes.*



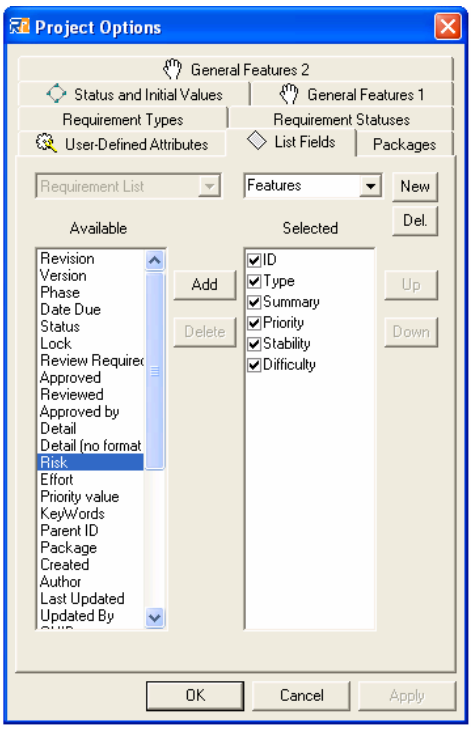
<fig. 6> Various requirements attribute

The easiest way to assign attributes to a requirement in RaQuest is to use the initially provided default values. The default priority levels are *high*, *normal*, and *low*, allowing immediate implementation without the need for further customization. <screen 10>

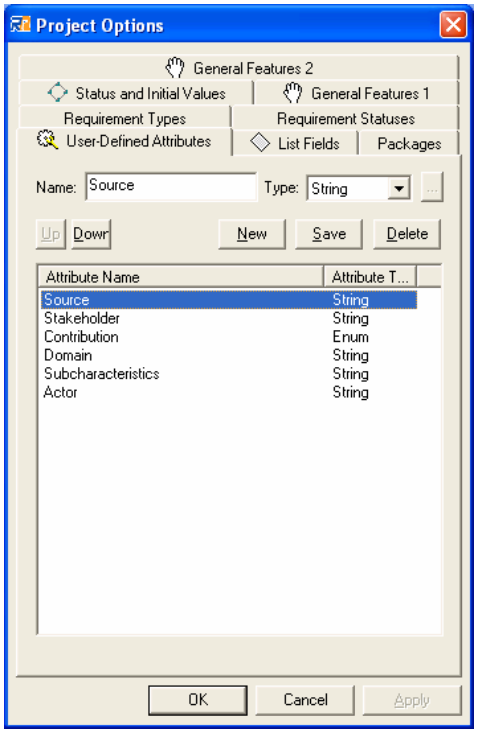


<screen 10> Attribute Settings by requirements' Details tabs

In actual projects, you will likely need to customize the list of requirement attributes. For example, you might want to change the *workload* attribute to *cost*, or change the priority rankings to *A*, *B*, and *C*, or add new attributes such as the existence of alternative approaches. To do this, use RaQuest's "User-Defined Attributes" screen to easily customize the list of provided attributes. <screen 11, 12>

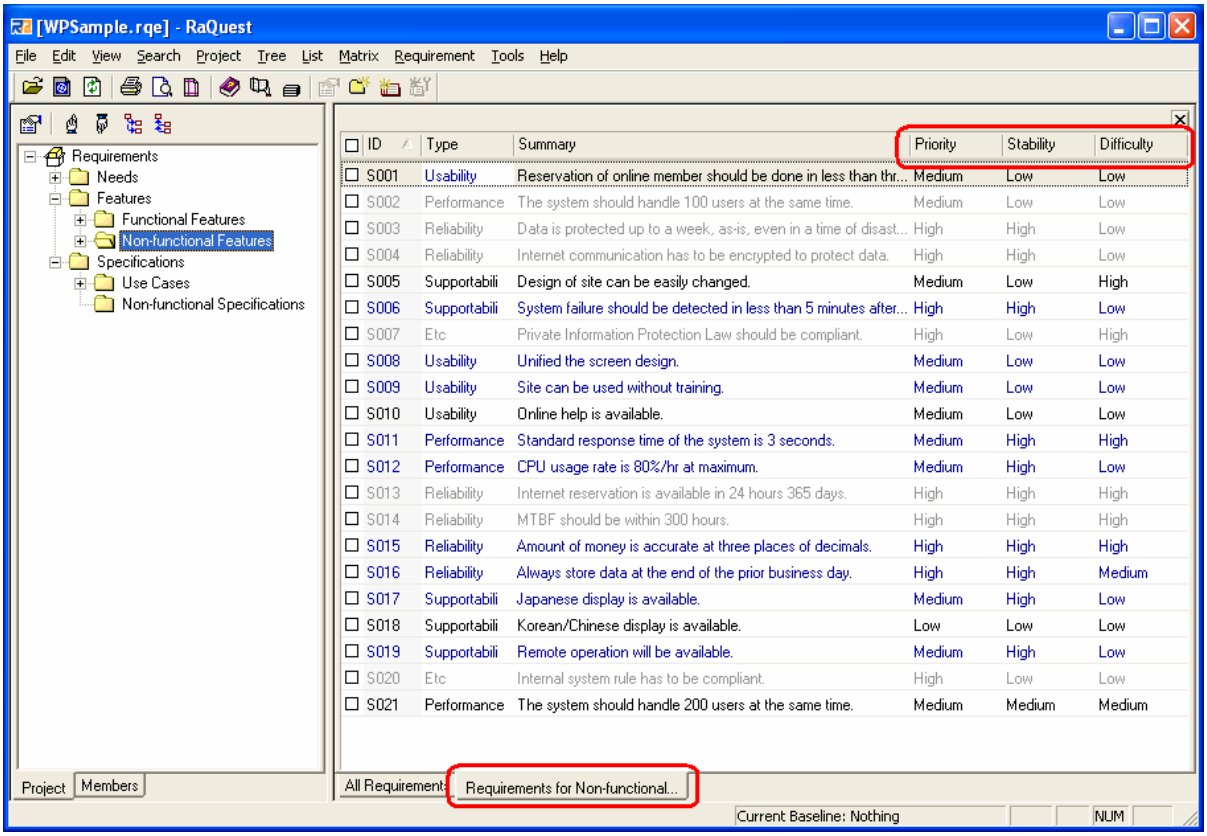


<screen 11> Creating User-Defined Attributes



<screen 12> Customization of List Fields

RaQuest furthermore facilitates defining requirement attributes by package. You might do this, for example, should you need a *difficulty* attribute in the Features package, but not in the Needs package. You can also set requirement attributes by package, just like changing a spreadsheet column type. The set of requirement attributes for packages can be viewed in requirement lists, as long as requirements are assigned to packages beforehand according to abstraction level and type.



<screen 13> Example of list differed by package

< Requirements management practice 4> Use requirement attributes to allow visualization of implicit knowledge

- Assign implicit knowledge concerning requirements as attributes
- Use requirement attributes suited to the project

<Practice 5> Control projects according to requirement status

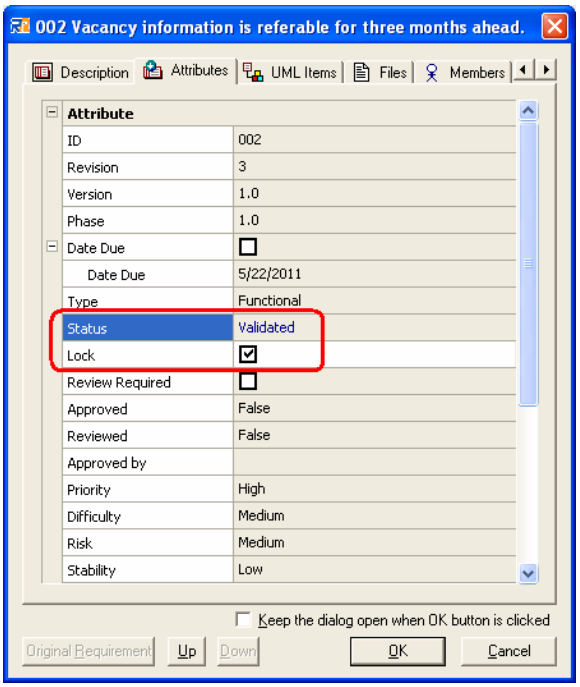
In addition to those attributes representing implicit knowledge, introduced as a part of Practice 4, it is also necessary to add an attribute representing requirement status. Visualization of project status allows for real-time comprehension of the state of requirements as they change. Requirement status attributes include the following:

Status refers to the current status of a given requirement in the context of the entire project..

Version *is an attribute that associates each requirement with baselines and system versions. It provides a mechanism for version control of requirement sets.*

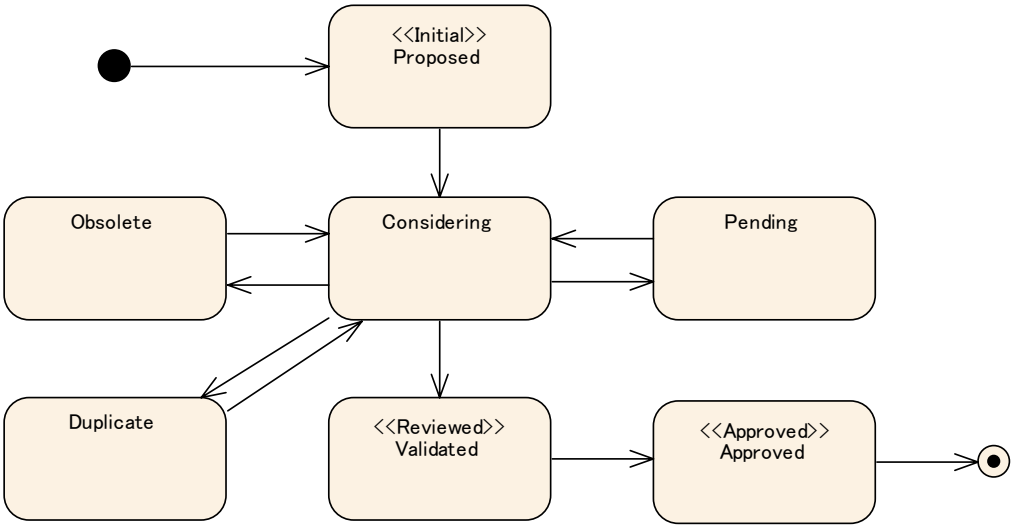
Revision *is an attribute assigned when requirements are changed as a record of changes. Many tools will automatically assign revision numbers.*

RaQuest includes features that allow the addition of status attributes for each requirement. Values can of course be updated by requirement, and triggers can be implemented that change requests when a specific status is assigned, allowing for the establishment of operational rules for tool-based project control. For example, requirements can be locked to prevent further changes when a *finalized* status is set. <screen 14>



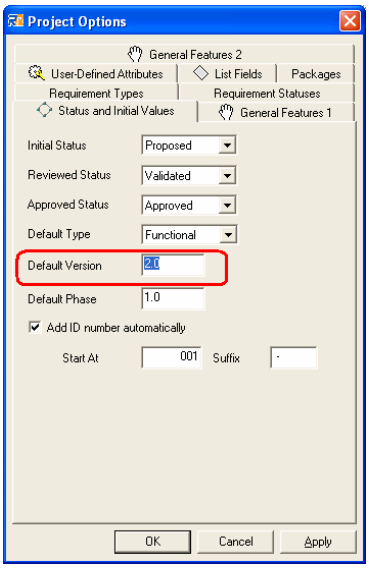
<screen 14> Validated requirements are un-editable

RaQuest was created under an assumption of requirements' state transitions like those shown in Figure 7, but this can be customized according to the rules of a specific project, just as status values can be changed.



<fig. 7> RaQuest's standard Requirements' state transition

RaQuest also supports *Version* and *Revision* status attributes. *Revision* values are automatically incremented each time a requirement is updated. *Version* attributes are generally used as a part of version control for entire requirement sets, and thus can be changed by batch under the project options. <screen 15>



<screen 15> Display the "Status and Initial Values" of Project Options

< Requirements management practice 5> Control projects according to requirement status

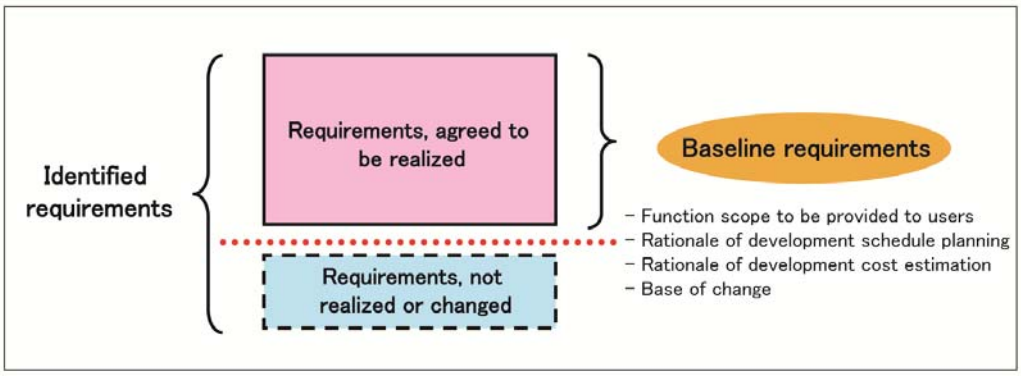
- Monitor and record requirement status
- Control project rules according to requirement status

<Practice 6> Use baselines to represent project scope

Requirement baselines are collections of requirements (requirement sets) agreed upon at the project outset. In project management, this is also referred to as the scope baseline. The IEEE Standard Glossary of Software Engineering Terminology defines “baseline” as follows:

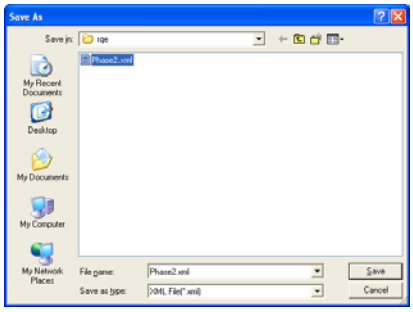
> *“A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change management procedures.”*

Requirement baselines are one of the most important concepts in the field of requirements management. Without baselines, it is impossible to know vital information such as the scope of features to be provided to the user, schedules, estimates, or the base of changes. Without clearly defining baselines, it is impossible to solve problems related to the ambiguity of software development scope.



<fig. 8> Baseline of requirements

RaQuest also includes requirements baseline management features. When determining project scope, it is possible to save baselines for identified requirement sets, including the status of each requirement at that time. Baselines can be saved either as external files or project files, and thus are saved as a first step when considering additions or changes to an already agreed-upon requirement set. This allows for review of saved baselines as an effective means of referencing their status during decision-making.



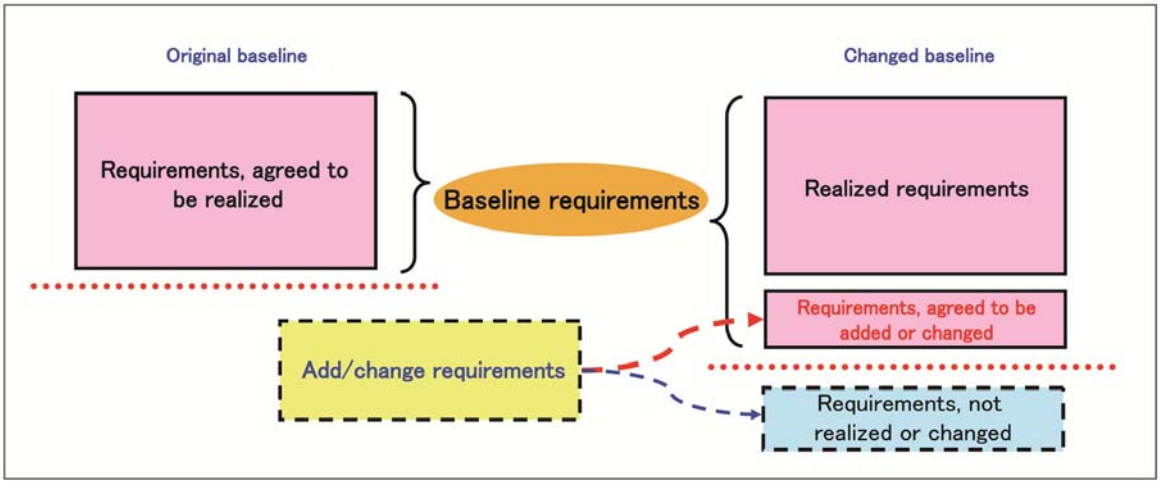
<screen 16> Save the baseline to an external file

< Requirements management practice 6> Use baselines to represent project scope

- Assign baselines to requirements
- Always maintain baselines in a manner that allows for referencing

<Practice 7> Maintain baselines through requirement change management

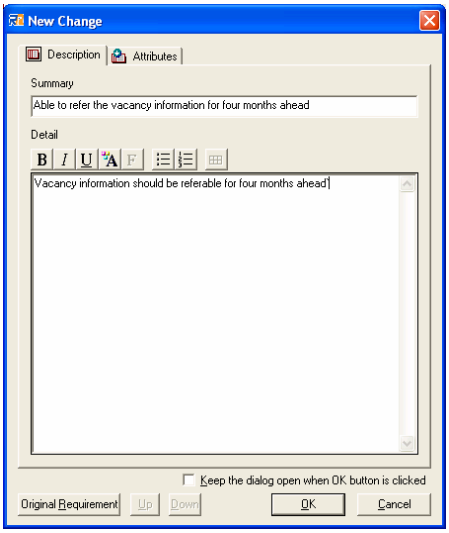
Changes to requirements can occur at any point during the project. Exactly what is taken to mean a “change,” however, has a significant impact on the project, and for fixed-term projects in particular, having agreed-upon rules is important for when changes take place. In projects where requirement baselines are being managed, controls should be established that allow changes to occur with regard to only baselines.



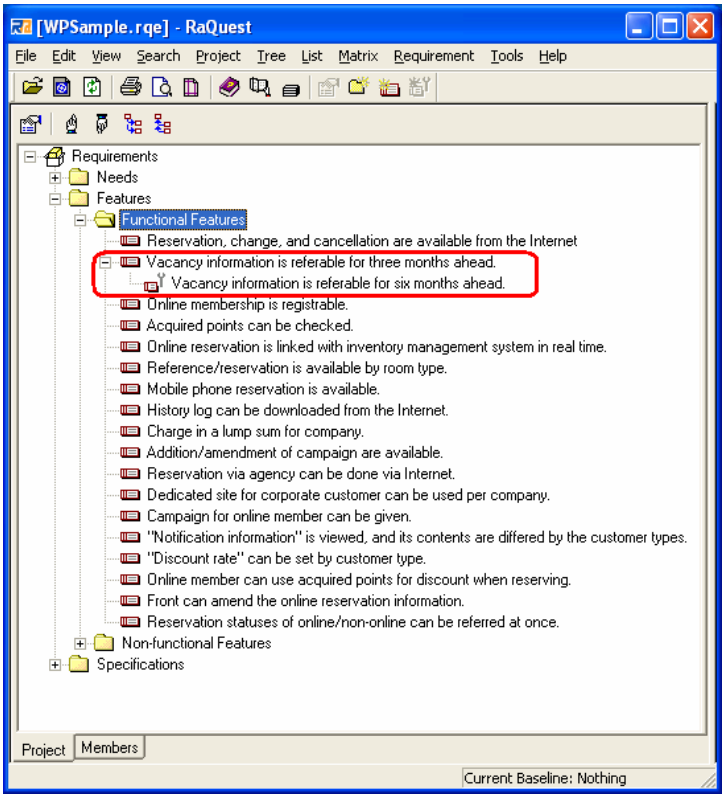
<fig. 9> Changed baseline

RaQuest is designed to allow changes to be applied to only agreed-upon requirements, a mechanism that allows for clear identification of changed requirements. As explained in the description of Practice 6, when requirements are agreed upon their status is set to *Approved*, and thereby establishing a baseline allows *Change* to be individually recognized as such. Requirement change in RaQuest has the following characteristics:

- New requirement change can be made for requirements with an only *Approved* status..
- A history of pre-change requirements is automatically saved.
- Changed requirements are identified as such in lists of requirements.



<screen 17> Creating new requirements



<screen 18> Check the change requirements by requirements list

< Requirements management practice 7> Maintain baselines through requirement change management

- Changes should be performed on requirement baselines
- Changed requirements should be displayed as such

4. Summary

Adaptation of requirements management to project

How do we adapt the methods of requirements management, introduced in this white paper, to an actual project?

Requirements management is a theme for the whole project, and it brings out the great effect only after it is applied to all teams. Individual operation by a specialist of requirements management is effective only to a small part. You need to have an ingenuity to spread out the effects to all teams. Here are the points for the effective use of requirements management mechanism to be adapted to project.

- Understand the advantages of requirements management, and share them with team members.
- Appoint a team member to be the person responsible for requirements management (someone dedicated to the task, if possible).
- Implement the mechanisms of requirements management from the start of the project.

Of course, another effective means of performing requirements management is the use of a tool like RaQuest. The following are some key methods for introducing RaQuest as a part of your requirements management solution:

- Identify those features of RaQuest that best fit with the style of requirements management for a given project.
- Use sample data to gain practice with the features you intend to use.
- Customize requirement attributes and status settings to best fit the characteristics of your project.

As discussed in the Introduction, requirements management can have a major impact on the success or failure of a development project, yet its methods have not yet taken firm root in real-life software development projects. We hope that you will consider the approaches introduced in this article, and by implementing them, see for yourself the transformation that your projects undergo.