UNIVERSITY OF NOVI SAD

FACULTY OF SCIENCES

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

RaQuest v3.1

Term paper from course Requirements Engineering

Professor:
Dr. Mirjana Ivanović

Student:
Robert Pap

Date:
10. June 2009 Novi Sad

Table of Contents

Abstract

Requirements management entails establishing and maintaining an agreement with the customer on the requirements for the software project. It contains several activities, like defining the requirements baseline, managing requirements changes, tracing individual requirements with other items, etc. Managing these activities can be extremely cumbersome especially for large projects, but their importance also grow with the size of the project. To help company workers with these activities, computerized and automated requirements management tools were introduced. One of the many software tools is *RaQuest* from *SparxSystems Japan* – the main subject of this paper.

This paper is organized as follows. The first chapter will explain the key concepts of requirements management and the idea behind the requirements management tools. The second chapter will introduce *RaQuest v3.1* by describing its key features. The chapter will continue with the usage of the tool in much more detail. Several screen shots will help the reader visualize the software as the chapter progresses. The chapter will use a practical case study, where a sample software's requirements will be entered into the tool.

Introduction

This chapter will describe the key concepts of requirements management and the idea behind the requirements management tools. This chapter is largely based on part I and part III of [Wiegers, 2003].

About Requirements Management

Requirements management "entails establishing and maintaining an agreement with the customer on the requirements for the software project" [Paulk et al., 1994]. The result of this agreement is the written Software Requirements Specification (SRS). The main activities of requirements management are:

- Defining the requirements **baseline**[1].
- Managing **requirement changes**, which includes reviewing of proposed changes, evaluating their impact on the whole project, and – if approved – incorporating these changes into the project itself.
- Controlling **versions** of individual requirements and their documents.
- **Tracing** individual requirements with other items, like other requirements, design or code elements, or test cases.
- **Tracking the status** of the requirements and thereby the status of the whole project.

Every requirement should have a set of **attributes** in addition to its textual description. An attribute is a piece of information attached to a requirement, adding to the context and background of that requirement. A list of possible requirement attributes is shown below[2]:

- Date the requirement was created,
- Its current version number,
- Author who wrote the requirement,
- Person who is responsible for ensuring that the requirement is satisfied,
- Owner of the requirement or a list of stakeholders (to make decisions about proposed changes),
- Requirement status,
- Origin or source of requirement,
- The rationale behind the requirement,
- Subsystem(s) to which the requirement is allocated,
- Product release number to which the requirement is allocated,

1 The requirements **baseline** is the set of functional and nonfunctional requirements that the development team has committed to implement in a specific release. The baseline cannot contain requirements that have been proposed but not accepted. Changes can be made only through proper channels.

2 Unless needed, don't add all of these attributes to the requirements. Selecting the smallest set of attributes will help manage the project effectively. Don't add an attribute if that will be empty for some requirements, or if that will duplicate some information already stored elsewhere.

- Verification method to be used or acceptance test criteria,
- Implementation priority,
- Stability (an indicator of how likely it is that the requirement will change in the future).

It's hard for developers to estimate, how much of the task is complete. They can think that they are 90% complete, only to realize that they were 90% complete a week ago, too. Also, another problem is that people define the word "complete" differently. Because of these reasons, it is better to replace the percentage style with a textual representation by defining the **status** of the requirement[3]. A list of several possible requirement statuses is shown below:

- **Proposed** – a requirement has been requested by an authorized source.
- **Approved** – the requirement has been analyzed, its impact on the project has been estimated, and it has been allocated to the baseline for a specific release. The key stakeholders have agreed to incorporate the requirement, and the software development team has committed to implement it.
- **Implemented** – the code that implements the requirement has been designed, written and unit tested. The requirement has been traced to design and code elements.
- **Verified** – the correct functioning of the implemented requirement has been confirmed in the integrated product. The requirement has been traced to test cases. The requirement is now considered complete.
- **Deleted** – an approved requirement has been removed from the baseline. It is recommended to include the reason of the deletion, and who made that decision.
- **Rejected** – the requirement was proposed, but is not planned for implementation in any upcoming release. Again, include the reason of the deletion, and who made that decision.

Software is considered to be prone to changes. A **change** to a requirement can happen practically anytime and from anybody: from customers and management to team developers, even testers. Letting requirement changes loose can be very dangerous: e.g. a change can turn out to be much harder to implement than expected. Because of this, before a change is accepted, its impact must be evaluated (this is called **impact analysis**). Also, a change can be proposed only through proper channels, otherwise the process will be full of "wild" changes: changes that are unknown to the majority of the team. Proposing a change through a proper channel will ensure that the change will not be overlooked, and that all team members will be properly informed about it.

The best way to avoid traps in change management is to document changes through a **change-control process**. A change-control process must be based on rules, called **policies**. Policies should be realistic, and they must be enforced. Some useful policies are:

- All requirement change requests shall follow the process. Requests submitted else-ways will be ignored.
- No design and implementation work until the change is approved.
- Requesting a change doesn't mean that it will be approved.

3 The agile method Extreme Programming (XP), is particularly sensitive to task completion status. In XP, there is a practice called "done done". A task is "done done", when everyone agrees it's done, that is: designed, coded, tested, integrated, installed, reviewed by a customer, and even documented. [Shore & Warden, 2008]

- The proposed change must be visible to all project stakeholders.
- Impact analysis must be performed for every change request.
- Every change to a requirement must be traceable to an approved change request.
- The rationale behind every approved or rejected change request must be recorded.

It is recommended to form a **Change Control Board** (CCB), that will manage the changes. The CCB can be one individual, or a whole group of people with different roles. Only this board can make decisions about change requests, update the request status and document them.

Evaluating the impact of a requirement change is probably the most important part of the change-control process. Trying to incorporate a change into the software can cause a ripple effect in various aspects: a change can be too expensive (in terms of money), or it can necessitate to change many documents, files or models. **Impact analysis** has three aspects:

1. Understanding the possible implications of making the change, e.g. adding a new functionality can degrade performances to unacceptable levels.
2. Identifying all the files, models and documents that will be modified because of the change.
3. Identifying the tasks required to implement the change and estimate the additional effort to complete the tasks.

Version control is an important aspect of requirements management. It is especially useful when changes happen. Version control will ensure, that all team members will have the most recent documents, including the requirements specification. Every version of a document must be uniquely identified. Every team member should access the current version of a document. Not only the version number must be available to a document, but the version control must also describe, what was changed (added, modified or removed) in comparison to the previous version of that document.

The last important part of requirements management is tracing the requirements with other items. As we already know, even simple requirement changes may require complex modifications. Because of this, an impact analysis is conducted before accepting the change request. The impact analysis is easier, if we know, how are the requirements linked to each other, and to other items, like design or code elements, or test cases. This is called **traceability**. Traceability information will help us identify all the work products we might have to modify to implement the requested change. It will help not only at impact analysis, but also in implementing changes to requirements and tracking the status of the requirements. Traceability can be also used to demonstrate that all requirements were implemented.

According to *Jarke* in [Jarke, 1998], there are three levels of abstraction: customer needs, requirements and product elements. Between these levels, he defined four types of requirements traceability links:

- Customer needs are traced **forward to requirements**, so we can tell which requirements will be affected if the customer needs change.
- To identify the origin of each software requirement, we can trace **backward from requirements** to customer needs. If the customer needs are represented as use cases, this will link the use cases and functional requirements.
- Tracing **forward from requirements** to product elements will ensure that all requirements are implemented.

- Product elements are also traced **backward to requirements**, which will answer to the question, why was each element created.

The most common way to represent the links between requirements and other items is in a **requirements traceability matrix** (also called a traceability table). This is a table with requirements as rows and other items (like other requirements, use cases, or product elements) as columns. Creating and maintaining these matrices is expensive and time-consuming, but it's very important, especially for large projects.

About Requirements Management Tools

Requirements management is not an easy task. Managing the activities can be extremely cumbersome especially for large projects, but their importance also grow with the size of the project. Even when a project team has a written SRS, it has numerous limitations:

- It's difficult to keep current,
- It's hard to communicate changes to team members,
- Tracing of requirements to other items is hard,
- Tracking the status of requirements is cumbersome, etc.

To help team members with these activities, computerized and automated **requirements management tools** were introduced. The main idea behind these tools is to store the requirements as a list, a table, or a database. For simple projects, a spreadsheet or a simple database will be sufficient to manage the requirements. For larger projects, a commercial requirements management tool is recommended. These tools can let the user to import requirements from source documents, define attribute values, export requirements in various formats, manage traceability links, track the status of requirements, etc. There are many reasons to use a requirements management tool. Some of them are[4]:

- Manage versions and changes,
- Store requirements attributes,
- Link requirements to other items (traceability),
- Track status,
- View requirement subsets – includes sorting and filtering of requirements depending of its attributes,
- Control access – deny access to unauthorized personnel, or define various permissions and restrictions,
- Communicate with stakeholders – notify affected personnel if a change has happened.

There are many classifications of these tools[5]. One of them is whether they are **database-centric** or **document-centric**:

- **Database-centric** tools store all requirements, attributes and traceability information in a database. Requirements can be imported from various source documents, but they then reside in the database.
- **Document-centric** tools treat a document created using a word-processing program (like *Microsoft Word*) as the primary container for the requirements. After importing the requirements from the source, these tools will store the requirements in a database. Once the requirements are in the database, we can define attributes and traceability links just like in database-centric tools. The main difference to database-centric tools is that document-centric

4 Note, that not every requirements management tool supports all of these features.

5 Today, the majority of requirements management tools try to add as many features as possible. This way, they can remain concurrent with the rest of the tools. The consequence of this trend is that the distinction between these classifications is starting to fade away.

tools will provide mechanisms to synchronize the database with the source documents.

Another (rather new) classification of requirements management tools is their **platform** (more precisely, whether they are **web-based** or not). The traditional approach was to install the tool on the computer (e.g. on *Windows*). Later, a client-server approach was introduced. Some tools have implemented even a web-interface, which would enable the team members to access the tool using a web-browser. Today, the **completely web-centric** requirements management tools are on the rise. These tools are controlled completely from web-browsers. They don't need any installation on computers, and the databases are stored on servers owned by the vendor of the tool itself.

These tools are rather expensive, but the high cost of requirements-related problems can justify the investment in them. Any of these tools will move the requirements management practices to a higher plane of sophistication and capability.

RaQuest v3.1 from SparxSystems Japan

RaQuest is one of the many requirements management tools. This chapter will describe this tool in greater detail. First, a general overview will help the reader to understand its key features including licensing information. After this introduction, the chapter will move forwards to the detailed description of this tool. The reader can treat this section as a summarized user manual. A sample case study will be used to illustrate the tool in action. This chapter is based on several official documents (mainly [SparxSystems RaQuest 3.1 Feature Guide, 2009], [SparxSystems RaQuest 3.1 Install Manual, 2009], [SparxSystems RaQuest 3.1 Startup Manual, 2009] and [SparxSystems RaQuest Help, 2009]) and personal experience.

Overview

RaQuest is one of the more popular requirements management tool available. It is developed by *SparxSystems Japan*.

About Sparx Systems and SparxSystems Japan

"**Sparx Systems** specializes in high performance and scalable visual modeling tools for the planning, design and construction of software intensive systems". [Sparx Systems, 2009] It is one of the leading vendors of solutions based on the UML (Unified Modeling Language). It is also a member of *OMG*[6] (*Object Management Group*, <http://www.omg.org/>). *Sparx Systems* was established in 1996 in Australia. It has a registered user base of close to 200.000 licenses. The majority of the user base is located in the US, Canada, UK, Spain, Germany, Japan, Scandinavia, France and the Netherlands. The flagship product of this company is **Enterprise Architect**. *Enterprise Architect* is a UML tool used for collaborative modeling, design and management. The most current version of this product (in June 2009) is 7.5. The official logo of the company can be seen on Image 1.

Image 1: The official logo of Sparx Systems

"**SparxSystems Japan** is one of the sister companies of *Sparx Systems*". [SparxSystems Japan Co., 2009] This company was established in Japan in 2002. It provides Japanese localized versions of *Enterprise Architect* and support for Japanese developers. *SparxSystems Japan* is also the main developer of *RaQuest*.

About RaQuest

RaQuest is an abbreviation for "Requirement Adjustment Quest" reflecting the hope of *SparxSystems Japan* that this tool will make system development easier. Interestingly, the pronunciation of "RaQ" means "easy" or "happy" in Japanese. The official logo of RaQuest can be seen on Image 2.

RaQuest is a very interesting specimen in the world of requirements management tools. The main reason behind this statement is that it requires another software product to run, and we cannot

6 *OMG* is an international, not-for-profit computer industry consortium. It is well-known for its enterprise integration, modeling and middleware (*CORBA*) standards.

classify it in a traditional sense, like other tools. To be more precise, it's difficult to say, whether it's database-centric or document-centric. It can parse a source document to import requirements from a SRS, and it will store the requirements in some kind of database, but the main reason of this confusion is the other software product it integrates to. *RaQuest* requires *Enterprise Architect* (*EA*) to run. As we know, *Enterprise Architect* is the flagship UML modeling tool of *Sparx Systems*. *RaQuest* can integrate itself with *EA* and the two programs will communicate seamlessly in both directions. *RaQuest* can import requirements directly from UML use case diagrams created in *EA*, and it can export requirements as use cases to use them in *EA*. Moreover, the extension of project files created with both programs is identical. Because of this interesting feature, it would be tempting to classify *RaQuest* as a **UML-centric** requirements management tool.

Image 2: The official logo of RaQuest

As for the platform, *RaQuest* is a **Windows-based** requirements management tool [Laatukonsultointi P. Kantelinen Oy, 2009]. It can connect to an *EA* server repository, so *RaQuest* has also some web functionality.

The minimal system requirements to run *RaQuest v3.1* are as follows [SparxSystems Japan Co., 2009]:

- *Microsoft Windows 2000* (32-bit, SP4 or later), *Windows XP* (32-bit, SP3 or later), *Windows Vista* (32-bit),
- *Enterprise Architect v5.0* or later required (*EA* must be installed prior to installing RaQuest),
- 20 megabytes of available hard-disk space,
- Screen resolution at least 1024x768,
- *RaQuest* won't run on Linux.

Features of RaQuest

With *RaQuest*, we can input summary and detailed attributes of a requirement. It enables us to list up requirements and export them to various formats (like *HTML* or *Microsoft Word*). Managing relationship information between requirements is also possible. *RaQuest* can display relationships of matrices useful for impact analysis. Because it is integrated with *Enterprise Architect*, it can create UML use cases from requirements or load them as requirements.

SparxSystems Japan advertizes *RaQuest* with the following official features:

- Print requirements lists,
- Export requirements lists as an *HTML* or a *Word* document,
- Show the relationship between requirements,
- Display relationships of requirements as a matrix, or of requirements and use cases,
- Generate use cases from requirements,
- Generate requirements from use cases,

- Track project personnel and assign requirements responsibility to them,
- Track changes of requirements and addition/deletion of changes.

Table 1 shows features of *RaQuest v3.1* collected by *INCOSE*[7] (*International Council on Systems Engineering*, <http://www.incose.org/>). *INCOSE* had created a requirements management survey site, sponsored by the *TDWG* (*Tools Database Working Group*). Together, they have developed a list of survey questions addressed to requirements management tool vendors. The vendors have sent back their responses, however, *TDWG* reserves the right to review and correct the answers in case of incorrectness. [INCOSE Tools Database Working Group, 2009] is probably the most complete and direct comparison of requirements management tools available today. Table 1 shows the survey responses specially for *RaQuest* answered by *SparxSystems Japan* and moderated by the *TDWG*. The responses were added on 24th June, 2005.

(see next page)

7 *INCOSE* is a not-for-profit membership organization founded in 1990 to "develop and disseminate the interdisciplinary principles and practices that enable the realization of successful systems". Its mission is to "advance the state of the art and practice of systems engineering in industry, academia and government". In December 2008, *INCOSE* had 6720 members. [International Council on Systems Engineering, 2009]

Question	Response	Response
1. Capturing Requirements/Identification	Full	No Response Added.
1.1. Input document enrichment/analysis	Full	RaQuest can import Requirement form existing documents (data) like EA project file, Word, CSV and also link any documents and graphics to Requirements.
1.1.1. Input document change/comparison analysis	Full	RaQuest does not compare documents themselves, but compare their time stamps. If RaQuest finds the difference between the time stamps, RaQuest alerts the user.
1.2. Automatic parsing of requirements	None	
1.3. Interactive/semi-automatic requirement identification	Part	mouse highlighting.
1.4. Manual requirement identification	Full	MS Word: Requirements are selected by the user in an MS Word document,and created using a right-click menu option. (Word Add-in). Requirements can also be created directly by the user.
1.5. Batch-mode operation	Full	RaQuest can import a lot of Requirements from CSV at a time. RaQuest compares their time stamps. If RaQuest finds the difference between them, RaQuest alerts the user, but does not update the link automatically.
1.6. Requirement classification	Full	Supports comprehensive requirements class typing with User Defined Attributes.
2. Capture System Element structure (if so, how? As document paragraphs? product structures?, …)	Full	No Response Added.
2.1. Graphically capture systems structure	Full	Yes. RaQuest can capture system implementation using requirement attributes or system implementation requirements that are linked to the requirements. Graphical traceability tree, matrix and UML chart show the relationships between system requirements and implementation.
2.2. Textual capture of system structure	Full	Yes. RaQuest can capture system implementation from any appropriate text documents and display the relationship textually between system requirements and implementation.
3. Requirements Flowdown	Full	No Response Added.
3.1. Requirements derivation (req. to req., req. to analysis/text)	Full	RaQuest generates requirements derivation automatically and display the relationship graphically between the source requirement and the derived requirements. RaQuest also monitors the update history. When the source requirement is updated, RaQuest alerts the user to reconsider the derived requirements.
3.2. Allocation of performance requirements to system elements (weight, risk, cost, etc.)	Full	RaQuest provides various system elements like Difficulty, Priority, Stability, and Risk to allocate performance requirements.
3.3. Bi-directional requirement linking to system elements	Full	Bi-directional link is fully supported in Matrix view. Matrix view has the ability to easily establish and abolish links between any type of object in any direction.
3.4. Capture of allocation rationale, accountability, test/validation, criticality, issues, etc. If so, how and what mechanism does it use.	Full	RaQuest provides several types of user-defined attributes. The number of user-defined attributes are unlimited. RaQuest provides the capability to assign any number of user-defined attributes to a group of objects. These attributes are easily reported on via Tree menu.
4. Traceability Analysis	Full	No Response Added.
4.1. Identify inconsistencies (orphans, ...) If so, what kind of...	Full	RaQuest has the ability to check inconsistencies among linked requirements and report it. It is easy to trace the cause of the inconsistencies.
4.2. Visibility into existing links	Full	RaQuest visually shows the relations among requirements. This

from source to implementation--i.e. follow the links		feature generates an EA diagram. RaQuest also shows the relations among requirements in matrix-style and user can easily edit the relations with the matrix diagram.
4.3. Verification of requirements (was it done, how was it done...)	Full	RaQuest has the ability to verify the fulfillment of requirements using Approve Feature. The approved requirement has the information of the approval. (Who? When? How?)
4.4. Requirement performance verification from system elements (roll up of actuals)	None	
5. Configuration Management	Full	No Response Added.
5.1. History of requirement changes, who, what, when, where, why, how	Full	RaQuest records update history of requirements automatically. Each requirement has its update log as the property. The update log shows who, what, when updated the requirement.
5.2. Baseline/Version control	Full	RaQuest has ability to generate Baselines as needed. RaQuest also compares the new and old Baselines and shows the difference between them.
5.3. Access control (modification, viewing, etc.)	Full	Enterprise Architect provides full access control to requirements.
6. Documents and Other Output Media	Full	No Response Added.
6.1. Standard specification output (if so, what kind)	Full	RaQuest has the wide documentation ability to output documentation in HTML, Word, CSV, and Excel format.
6.2. Quality and consistency checking (spelling, data dictionary,...)	Part	RaQuest has the powerful glossary feature to manage terms used in the project. The glossary feature is completely bidirectional with the glossary feature in Enterprise Architect.
6.3. Presentation output	Full	RaQuest generates EA diagrams. The diagram is compatible with Enterprise Architect. The user can edit or print it in Enterprise Architect.
6.4. Custom output features & markings (definable tables, security marking)	None	
6.5. WYSIWYG previewing of finished output	Part	RaQuest has the ability to allow the user to preview Requirement List on-screen in finished format before its printing, but generated document like Word cannot be previewed on-screen before its generation.
6.6. Status Reporting	Full	RaQuest has Status Counter feature. It gathers up the status information of the project and saves the information. The saved status report can be exported in CSV format.
6.6.1. Technical Performance Measurement status accounting	None	
6.6.2. Requirement progress/status reporting	Full	Filtered information can be used as the status reporting on current compliance/non-compliance to various requirements.
6.6.3. Other ad hoc queries & searches	Full	RaQuest provides ad hoc queries and searches. The user can combine various search conditions.
7. Groupware	Full	No Response Added.
7.1. Support of concurrent review, markup, & comment	Full	RaQuest provides a multi-user solution for any number of users sharing the project information. RaQuest allows all users access to the same requirements at the same time. The automatic lock systems is activated when multiple users access to the same requirements at the same time to avoid the inconsistencies.
7.2. Multi-level assignment/access control	Full	Multi-level assignment/access control is enabled by User Permissions feature of Enterprise Architect. (for only users of Enterprise Architect Corporate Edition)
8. Interfaces to Other Tools	Full	No Response Added.
8.1. Inter-tool communications	Full	RaQuest works with UML tool Enterprise Architect. The project

		file (data file) is completely compatible for both RaQuest and Enterprise Architect.
8.1.1. Interfaces to other tools?	Full	RaQuest has interfaces to the following tools: • Enterprise Architect (data compatibility) • Microsoft Word (Requirements Import/Export) • Microsoft Excel (Requirements Import/Export)
8.1.2. External Applications Program Interface available	Part	Enterprise Architect provides external applications program interface as the add-in feature.
8.1.3. Support Open database system (standard query access)	None	
8.1.4. Import of existing data from various standard file formats	Full	Yes. RaQuest can import data from the following file formats.* Enterprise Architect project file* CSV format* Microsoft Word (add-in feature)
8.1.5. Support Data Exchange Standards (AP-233, XML,..)	Part	Requirements baselines are saved in XML format.
8.2. Intra-tool communications	Full	No Response Added.
8.2.1. Exchange of information between same-tool different installations	Full	Data can be exchangeable in XML format. Users can open the same project at the same time. And users can use the same project at the same time, when the project is located in database.
8.2.2. Consistency/comparison checking between same-tool datasets	None	
9. System Environment	None	
9.1. Single user/multiple concurrent users	Full	RaQuest support both a single user and multiple concurrent users.
9.2. Multiple Platforms/Operating Systems?	Full	Microsoft Windows2000(x32 SP4)/XP(x32 SP2)/Vista(x32)
9.3. Commercial vs. Proprietary database	Full	RaQuest can be used with commercially available database like Oracle and SQLServer.
9.4. Resource Requirements	Full	"[Software] Enterprise Architect 5.0 (or later version) [Screen] 1024x768 screen resolution preferred for showing lists and manipulate Requirement items - 800x600 useable, 640x480 not suitable. "
9.4.1. Memory requirements	Full	[RAM] depends on OS - 96 MB for Windows 2000, Windows XP and Windows Vista, (more RAM will improve performance).
9.4.2. CPU Requirements	None	
9.4.3. Disk space requirements	Full	20MB of available hard-disk space
10. User Interfaces	Full	No Response Added.
10.1. Doing one thing while you are looking at another	Full	Yes. RaQuest always shows Tree and List. Tree shows the hierarchical structure of Requirements and List shows the summary of Requirements. You can open the properties dialog of a Requirement at the same time.
10.2. Simultaneous update of open views	Full	Yes. The change automatically reflect in all other views, when it is saved.
10.3. Interactive input/control of data	None	
10.4. Which window standard do you follow?	Full	Microsoft
10.5. Executable via scripts (recordable) or macros	Part	Yes. The user can create add-in programs in Enterprise Architect.
10.6. Web browser interface?	Full	Requirements can be exported to HTML.
10.7. Edit Undo Function Support	Part	attribute can rollback used update log.

11. Standards--which one(s) do you comply with?	Full	RaQuest supports many output formats/standards. It is currently used in various industries like electric, automobile and IT industries.
12. Support and Maintenance	Full	No Response Added.
12.1. Warranty	Full	Support for a full year.
12.2. License policy (Network, Node Locked,..)	Full	Node Locked License or Specify User License or Floating License.
12.3. Maintenance & upgrade policy	Full	Basically every two weeks. However we respond fatal bugs immediately as needed. Registered user can use the latest version for a year at no additional fee.(subscription period) At the end of subscription period, registered users can select whether they renew the subscription or not. If they choose the renewal, they can keep on using the latest version. If not, they lose the right to download the latest version, but they can keep on using the product which they have now.
12.4. On-line help	Full	Everyone can download the manuals and latest HTML help file from our web site. We also provide email support service.
12.5. Internet access/Website	Full	Visit http://www.raquest.com/
12.6. Phone support	None	
12.7. Support users group	Full	Everyone can use User Forum. (Internet Discussion Board)
13. Training	Part	Only in Japan.
13.1. Tool-specific training classes	Part	Startup Manual and Install Guide describe its powerful usability. They can be downloaded from our web site freely. Tool-specified training class in Japan.
13.2. Training available at customer's location	Part	Contact us if users would like to have a training in Japan, especially near Tokyo area.
13.3. Recommended training time	None	There are no official data about the recommended traning time, but the tool is highly user-friendly.
13.4. Software installation with only basic training	None	The installation is very easy and the plain installation guide is sent to a registered user with the delivery of license key by email.
14. What other requirements management features do you as a tool supplier think are important (modeling, etc...)?	None	

Table 1: Features of RaQuest v3.1 according to INCOSE

Licensing Information

Normally, the requirements are managed by a specific staff or a part of the team, because requirements management tools are generally too expensive to distribute it to all members of the team. *RaQuest* is designed to change that development style. This tool is cheap enough to be distributed to all members, so all members related to the development can share the information. This development style is one of the goals that *SparxSystems Japan* aims for.

RaQuest can be evaluated using a trial version for 30 days[8] [SparxSystems Japan Co., 2009]. After this period, a valid license must be purchased. A purchased license is valid for 12 months. During this period, the registered user can download new and full versions or updates of *RaQuest* for free, he can access the *SparxSystems Japan*'s email support, and he can access the exclusive registered users' area. After this period, the user can renew the license and enjoy the subscription benefits for another year. If the user decides to not renew the license, he can continue to use the current version of *RaQuest* without any limitations (and he can reinstall it, so it is recommended to

8 Because *Enterprise Architect* is needed to run *RaQuest*, a 30-day trial version of *EA* is also available.

archive the installation files for later use), but without the subscription benefits: he won't get exclusive technical support, and he won't be able to download newer versions of the software. If the user wants to get newer versions, he'll need to renew the license.

There are two main types of licenses available to *RaQuest*: **node-locked** and **floating**[9]. A node-locked license permits the user to use the licensed software on a specific machine (when the license is installed, it'll reside on that machine). A floating license can be allocated dynamically to different machines on a network (a machine can receive a license when the software is started, and the license will be revoked when the user closes the software). When a company buys a number of floating licenses, this number represents the number of concurrent usage of that software. This means, that the number of machines, which can use the software, can be much bigger then the number of purchased licenses, as long as there are no more computers using the software simultaneously than the number of licenses. The price of licenses (effective on June 2009) are given in Table 2. Note, that prices can be higher if the company doesn't own *Enterprise Architect*.

Type of license	Number of licenses	Price per license (new)	Price per license (renewal)
Node-locked	1 to 4	150 USD	50 USD
	5 to 19	145 USD	48 USD
	More than 20	140 USD	46 USD
Floating	1 to 4	220 USD	75 USD
	5 to 19	210 USD	70 USD
	More than 20	200 USD	65 USD
Note: a node-locked license can be upgraded to a floating license. In this case, the price of upgrade is 70 USD for each license.			

Table 2: Licensing information for RaQuest (June 2009)

9 An academic license is also available.

Installing RaQuest v3.1

Installing *RaQuest* is very easy. First, the user needs to download and install *Enterprise Architect* before proceeding to the installation of *RaQuest*. In this term paper, the trial version of *EA* and *RaQuest* was installed (all installation files can be accessed from the download page at <http://www.raquest.com/products/raq_downloads.htm>). Installation of *EA* and *RaQuest* is pretty straightforward. During the installation of *RaQuest*, the only interesting part is the feature or component selector screen shown at Image 3. The only selectable extra feature is the "*Word Addin*". This add-on allows the user to add requirements to *RaQuest* directly from an opened *Microsoft Word* document. As we can see, the installation of this feature is disabled by default, but we will install it for demonstration purposes.

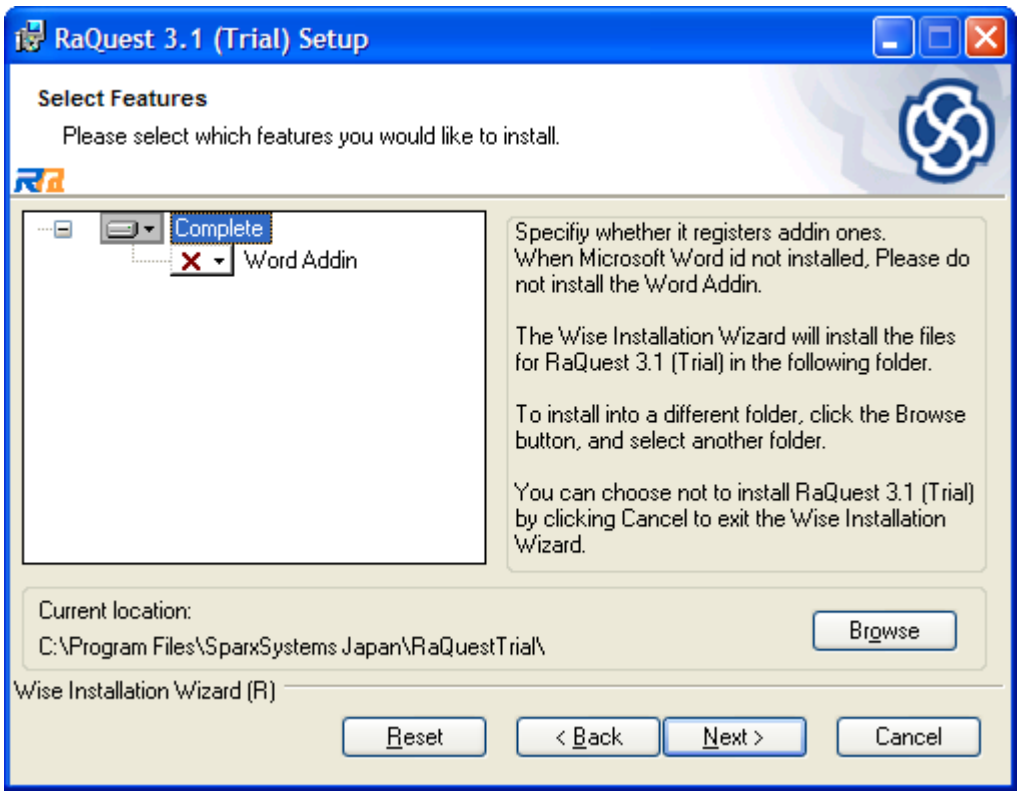

Image 3: The feature selector screen during the installation of RaQuest. The Word Addin feature is disabled by default.

After the installation, the trial version of *RaQuest* is ready for use for 30 days.

Discovering RaQuest v3.1

When we open *RaQuest*, a small trial window will appear first that notifies us about the number of days left until the trial expires. When we click on OK, the main window of *RaQuest* will be shown (Image 4). The main window consists of two panes. The left one is the **tree pane**, which shows packages and requirements in a tree-like view (by default). The right pane is the **list pane**, which shows the list of requirements (by default). It is important to understand that both panes are working as tabs. This means, that the user can set a different view to a pane (e.g. display the tree of members in the left pane instead of packages, of list comments instead of requirements in the right pane), but the old view will be saved in a tab. Later, when the user would wish to return to the old view, he just needs to click on the corresponding tab. The tabs are shown in the lower part of the panes.

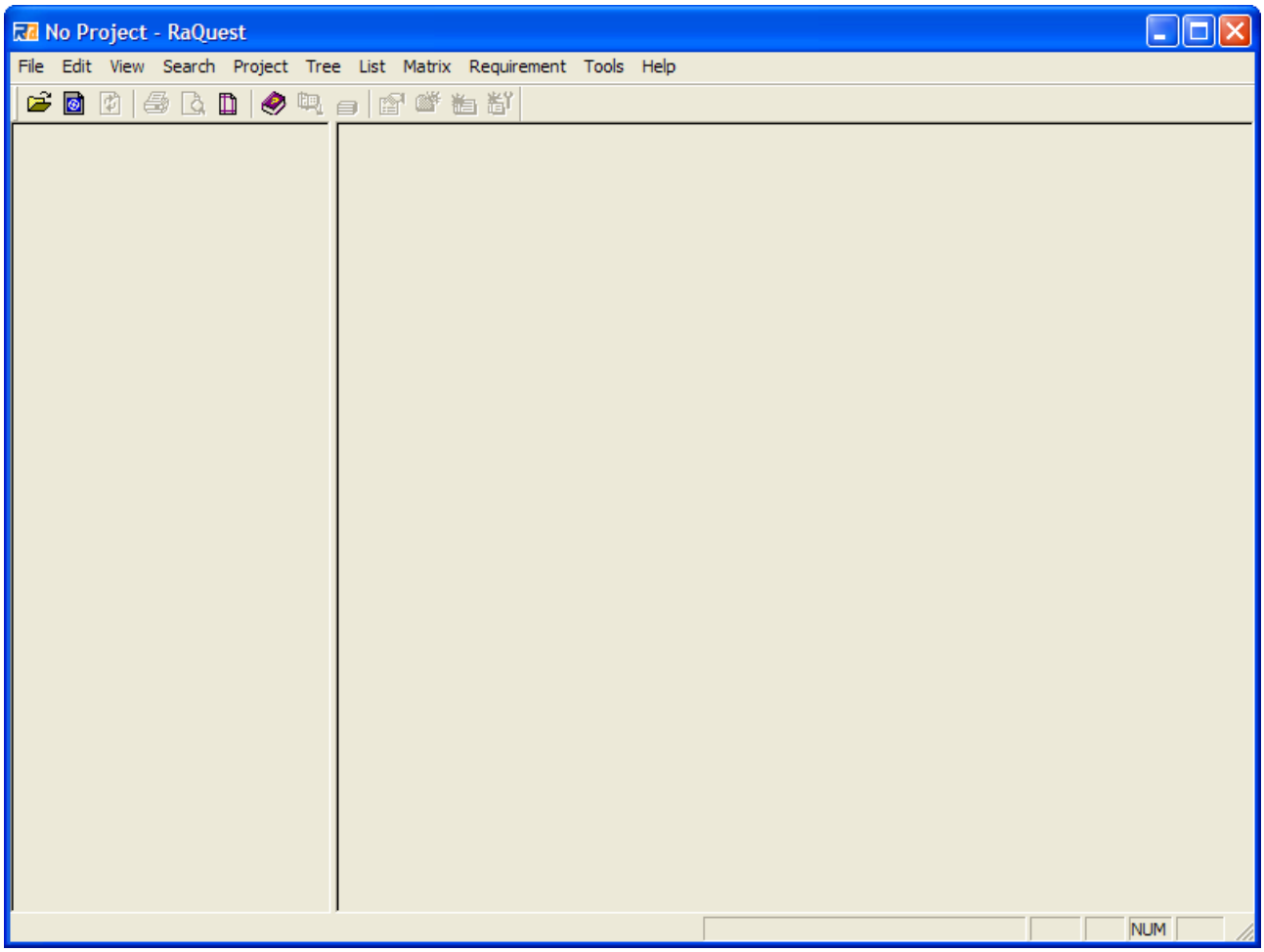

Image 4: Main window of RaQuest directly after start-up

First, we will describe the menu system.

The Menu System

This section will describe the menu system of *RaQuest* (Image 5). Some of the commands will be explained in more detail in later sections of this paper.

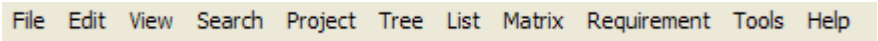

Image 5: The menu system of RaQuest

The **File** menu contains several commands which are common in all *Windows*-based programs (like New, Open, Close, Print or Exit), but it contains several unique commands, too. The commands are as follows:

- **New** – creates a new *RaQuest* project file.
- **Open** – opens an existing project file. *RaQuest* can open *RQE* (*RaQuest* specific project) and *EAP* (*Enterprise Architect Project*) files.
- **Connect to Server Repository** – connects to an *Enterprise Architect* server repository. Clicking on this command will show a new window, where the user will be asked to enter the connection string for the repository. If *EA* is also connected to this server, the two programs will communicate across a network.
- **Close** – closes the current project file.
- **Reload** – this command will close and reopen the current project file. Useful if the project file is used simultaneously by more than one user, because it will reflect the changes made my the other users.
- **Print** – prints the current list which is currently showed on the right pane. If the user wants to print out only specific items, he can use the Search or Filter commands first.
- **Print Preview** – displays the current printable list on the screen before printing.
- **Print Setting** – displays the current printer settings. This command is also accessible from the **Print** command.
- **Import** (sub-menu) – imports an external file and adds requirements from that file to the package which is currently selected. Only files with the *CSV*[10] file extension can be imported.
- **Export** (sub-menu) – this command will export the requirements to various file formats: *HTML*, *Word*, *RTF*, *CSV* and *Excel*.
- **Collect Related Files** – copies all files which are associated with the requirements to the specific folder. This command is useful if the user wants to migrate the project to another computer.
- **Exit** – closes *RaQuest*.

The **Edit** menu shows commands common in other *Windows*-based programs. These are:

- **Copy** – copies a selected requirement to the clipboard.
- **Cut** – copies a selected requirement to the clipboard, and deletes it from the list.
- **Paste** – pastes the requirement copied or cut earlier. If a package is selected, the requirement will be pasted under the selected package. If a requirement is selected, the requirement will be pasted as a child requirement of the selected requirement.
- **Delete** – deletes the selected requirement from the list. A confirmation dialog will be shown before deletion.

10 A *CSV* file (an abbreviation of ***C****omma* ***S****eparated* ***V****alues*) can be opened in many programs, including *Microsoft Excel*.

The **View** menu shows the following commands:

- **Toolbar** – shows or hides the toolbar (the upper list of buttons with icons),
- **Status Bar** – shows or hides the status bar (the lower part of the main window),
- **TreeToolButtons** – shows or hides the upper toolbar of the left pane (tree pane),
- **ListToolButtons** – shows or hides the upper part of the right pane (list pane).

As the name suggests, the **Search** menu allows the user to search or filter in the requirements list. The menu commands are as follows:

- **Search** – finds the first requirement which satisfies a given condition. When the user clicks on this command, the search dialog will be shown, allowing the user to define search criteria.
- **Search Next** – moves to the next requirement which satisfies the condition given previously in the Search dialog.
- **Search Prev.** – moves to the previous requirement which satisfies the condition given previously in the Search dialog.
- **Filter List** – the Search dialog allowed the user to search for a specific requirement. The **Filter List** command allows the user to find all requirements which satisfy a given condition. The resulting list can be shown in the current list tab, or in a new one. Note, that this command can filter not only requirements, but UML items, files, assigned members, test items and even comments.
- **Replace** – searches for a specific text string and replaces it with the given string. *RaQuest* can search for text strings in the requirement summaries, details, and in the package name.
- **Search in Tree** – this command is similar to the Search command, but searches in the project tree (not in the requirements list).

The **Project** menu contains commands specific to a project, like version-control, showing project statistics, etc. The commands are:

- **Manage Baselines** – manages baselines. Note, that a “baseline” in *RaQuest* is not equal to a “baseline” defined earlier in this term paper. In *RaQuest* terminology, a baseline is the status of a project in a specific moment. It is like saving a snapshot of a project in a specific moment. Later, the user can compare the old snapshot with the current status to find differences (and to see, how the project evolved during that time period). When the user clicks on this command, he can save a snapshot of the project, compare it to an earlier snapshot, rollback to an earlier snapshot, or even branch a baseline.
- **Glossary** – every project needs a glossary. A glossary is a set of terms with their definition (description) used in the specific project. *RaQuest* allows the user to distinguish the technical terms from the business terms.
- **Security** (sub-menu) – this sub-menu has two commands: **Change Login User** and **Change Password**. If *EA* is used as a source project, the user can define various users with various permissions (in *EA*). If such a project is opened in *RaQuest*, the user will need to enter the user-name and password to access the project. If the user wants later to login as a different user, he can use the **Change Login User** command. Note, that these commands are disabled

if no users were defined in *EA*. (The **Change Password** command is disabled in this version of *RaQuest*.)

- **Statistics** (sub-menu) – this sub-menu has two commands: **Display Current Status Report** and **Output Report as CSV**. The first one will display some statistics for the current project, like the number of requirements, files, assignments, comments, number of functional requirements, and number of proposed requirements. The second command will export this report to a *CSV* file.
- **Duplicate ID Check** – this command will check for duplicate requirement Ids.

The **Tree** menu allows the user to manipulate with the left pane. When the user clicks on a different tree view, it will be opened in a new tab. Five commands are available:

- **Project Tree** – displays the project tree. This is the default view of the left pane.
- **Member Tree** – displays the member tree in the left pane (*RaQuest* allows us to define members).
- **by Type** – displays a tree sorted by the types of requirements. The types are: “Functional”, “Performance”, “Printing”, “Report”, “Testing”, “Validate” and “Display”.
- **by Status** – displays a tree sorted by the combination of requirement version and status. The root (or roots) of the tree will be the version number itself. The possible statuses are: “Proposed”, “Approved”, “Mandatory”, “Validated” and “Implemented”.
- **Custom Tree** – displays a new dialog, where the user can choose to sort items by any attribute.

The **List** menu allows the user to manipulate with the right pane. When the user clicks on a different list view, it will be opened in a new tab. This is the longest menu. It consists of 17 commands:

- **Requirement List** – displays the list of requirements. This is the default view of the right pane.
- **Change List** – displays changes associated with the selected item.
- **To-Do List** – displays the list of incomplete requirements. This view will list all requirements which don't have the “Reviewed” or “Approved” status.
- **Unassigned Requirement List** – displays a list of requirements which weren't assigned to a member.
- **Pending Approval List** – displays the list of requirements which were validated, and are waiting for approval. This is handy if a member of the CCB wants to approve the pending requirements easily.
- **List with Priority Value** – lists requirements sorted by their priority. The priority is calculated from the “Priority”, “Difficulty” and other attributes. Approved and validated requirements are excluded from this list.
- **Update Log List** – this view lists the change-log of the project. All updates are listed by their date and time, user who did the update, and what was updated.
- **Package List** – displays the list of packages for a project.

- **Member Requirements** – displays requirements assigned to a selected member. A member from the member tree must be selected before clicking on this command.
- **Relationship List** – displays a list of relationships between requirements. Useful for showing traceability information. The list will be empty if no relationships were defined.
- **Destination Requirements** – displays a list of requirements which will be affected by the modification of the selected requirement. Useful for impact analysis. The user just needs to click on a requirement, and all linked requirements will be shown.
- **Source Requirements** – displays a list of requirements whose modification will affect the selected requirement. This command is the opposite of the previous command (Destination Requirements).
- **UML Item List** – shows the list of UML items which are linked to the selected package or requirement. If a requirement is selected in the left pane, this view will show all UML items associated with that requirement. If a package is selected, the list will contain all UML items which are associated with all requirements of the selected package.
- **File List** – shows the list of all files associated with the selected package or requirement.
- **Assigned Member List** – displays a list of members assigned to the selected requirement. If a package is selected from the left pane, this command will show all members assigned to all of the requirements of the selected package.
- **Test Item List** – we can also assign test cases to requirements for the testing phase. This command will show the list of all test items for a selected requirement. A package can also be selected.
- **Comment List** – displays comments for a selected requirement. A package can also be selected.

The **Matrix** menu can be used to generate traceability matrices. The menu consists of the following commands:

- **Matrix** – opens the Matrix window. The user must select the matrix type first, and then click on the Update button. The user can export the matrix to a *CSV* file. The available types of traceability matrices are: “Requirements”, “Member”, “Use case” and “UML item”. The other four commands of the Matrix menu are included just to speed up this process.
- **Requirements Matrix** – opens the Matrix window, and displays the requirements matrix immediately.
- **Member Matrix** – opens the Matrix window, and displays the member matrix immediately.
- **Use Case Matrix** – opens the Matrix window, and displays the use case matrix immediately.
- **UML Item Matrix** – opens the Matrix window, and displays the UML item matrix immediately.

The **Requirement** menu can be used to manipulate with requirements. The following commands are available:

- **Properties** – displays the Properties dialog for a selected requirement. Here, the user can define various attributes, assign the requirement to members, update the status, link it with

UML items, files and test items, write a comment, etc. (Note, that the Properties dialog can be accessed by double-clicking on the requirement itself, or by right-clicking on the requirement, and selecting Properties from the drop-down menu).

- **New Package** – creates a new package under the selected package or requirement (as a child element).
- **New Requirement** – creates a new requirement under the selected package or requirement (as a child element).
- **New Change** – defines a change to the selected requirement. The change will appear as a new requirement, but it will be linked to the original requirement. Note, that this command is available to approved requirements only.
- **New Comment** – adds a new comment to the selected requirement.
- **Approval** – approves the selected requirement. Note, that only validated requirements can be approved.
- **Set Constraints between Requirements** – constraints can be defined to automate the validation of requirements. The constraints can be defined between two requirements: "and" (the requirement will be validated automatically when its related requirement is valid) and "or" (the requirement will be invalidated automatically when its related requirement is valid). (This command is disabled in this version of *RaQuest*.)
- **Set Relationships between Requirements** – sets the relationships between requirements. (This command is disabled in this version of *RaQuest*. The drag-and-drop methodology is used instead.)
- **Show Relationships Map** – displays a visual representation of relationships between the selected requirement and other requirements.
- **Show Change Effect** – displays visually the extent of an effect caused by the modification of the selected requirement.

The **Tools** menu contains some extra features of *RaQuest*. The commands are as follows:

- **Run Enterprise Architect** – starts *EA*, and automatically loads the current project in it.
- **Enterprise Architect features** (sub-menu) – the following commands are available:
  - **Generate Use Cases** – generates use cases from requirements. Useful for *EA*.
  - **Generate Classes from Glossary** – *RaQuest* can also generate classes from glossary terms. This information can be useful in relationship-analysis.
  - **Import from this Project** (sub-menu) – this sub-menu contains two commands: **Generate Requirements from Use Cases** and **Import Internal Requirements**. The first one will generate requirements from use cases of the current project. This command is mandatory, if the user has opened an EA project, and wants to transform the use cases to requirements. The seconds command will generate requirements from internal requirements. Internal requirements are requirements defined in EA use cases. If the user decides to import those requirements, then he will choose this command (instead of the first).
  - **Import from other Project** (sub-menu) – this sub-menu has two commands: **Requirements** and **Use Cases**. The first one will import requirements from another (not

currently opened) *EA* project, and the second will import use cases from another *EA* project and transform them into requirements. (Both commands are disabled in this version of *RaQuest*.)

- **Run Execution Log Viewer** – shows the execution log of *RaQuest*. This command is used for debugging purposes.
- **Project Options** – shows the Project Options dialog. This dialog contains several settings for the currently opened project. Here, the user can define new user attribute types or statuses, assign whole divisions to requirements (by default, only individual members can be assigned), prevent the deletion of existing requirements, etc.
- **Local Options** – shows the Local Options dialog. This dialog contains several settings for *RaQuest* itself. Here, the user can customize the user interface of *RaQuest* (like coloring of requirements based on their statuses, etc.).

Lastly, the **Help** menu is self-explanatory. The following commands are available:

- **About RaQuest** – the About dialog is shown (version of *RaQuest*, and licensing information).
- **Input Updated License Key** – displays a dialog to enter an updated license key. This command is disabled in the trial version of *RaQuest*.
- **Help** – opens the integrated help system.
- **RaQuest Website** – opens the official *RaQuest* website in the default web-browser.
- **Check for Newer RaQuest Build** – connects to the internet to check for a newer version of *RaQuest*.

In the next section, we will present some basic actions performed in *RaQuest*.

Basic Operations

In this section we will describe the basic functions of *RaQuest*, e.g. how to create a new project, define a requirement with attributes, assign them to members, etc.

First, it is important to understand, that there are three different **requirement items** in *RaQuest*:

- **Requirements** – the ordinary requirements. In *RaQuest*, all requirements must be child elements in the hierarchy. In other words, all requirements must have a parent: a package or an other requirement. If a requirement has another requirement as its child, we say that they are linked (relationship).
- **Packages** – these items serve as containers of requirements. When a project is created, a root package will be made. All other items (requirements, packages, etc.) will be placed under this root package. Packages can have multiple purposes. They can help us manage large numbers of requirements, categorize requirements, but they can also serve as a realization of project baselines (in the traditional terminology, not in the *RaQuest* terminology). For example, we can define a package, and store postponed requirements in there (requirements which were rejected in the current baseline, but they will be reconsidered in a later stage of the project). Note, that a requirement can also have a package

as its child. In the tree pane (left pane), the user can freely drag-and-drop requirements and packages. The following moves are legal: drag a requirement to another requirement, drag a requirement to a package, and drag a package to another package. Dragging a package to a requirement is not allowed.

- **Changes** – these are the changes to requirements as defined in the traditional terminology. In *RaQuest*, a change is a special requirement for a requirement to be changed. Instead of rewriting (modifying) an existing requirement, *RaQuest* will define a new requirement, and link it to the old requirement as its child. It will also replace the icon of the requirement to distinguish the link from ordinary relationships. This methodology is very smart, because we will be able to recognize the full history and the changes of a specific requirement.

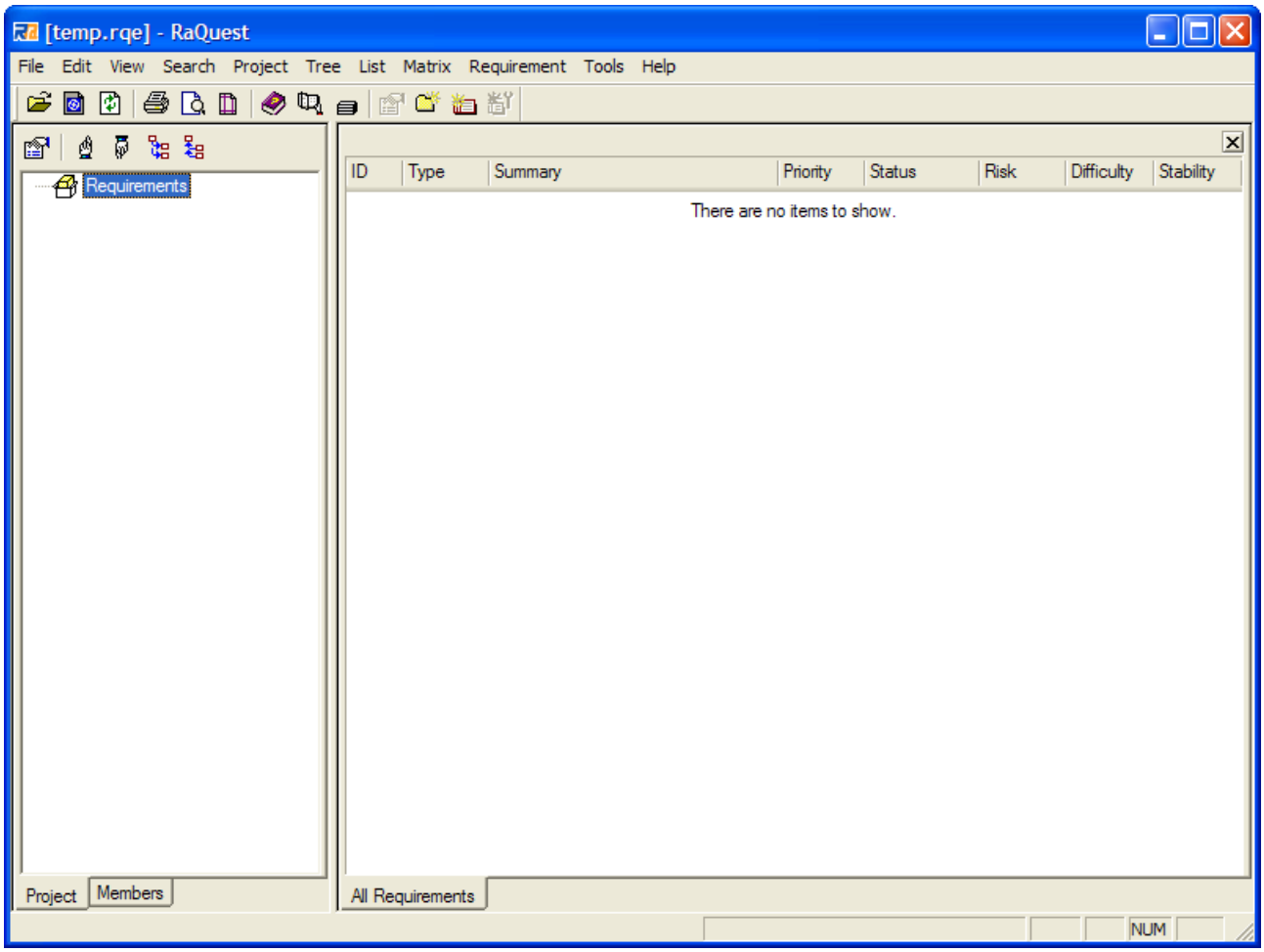

Image 6: Main window of RaQuest after a new project has been created

We will now create a new project and define there a new package and a requirement[11]. To create a new project, open the File menu, and choose **New**. *RaQuest* will ask us to choose a location on the hard drive for the project. When we click on Save, *RaQuest* will link itself with *Enterprise Architect*, and it will create the root package named as "Requirements" (Image 6). We can set the properties of this root package by double-clicking on it[12] (Image 7).

11 We will use a fictitious development of a Course Registration System as an example. This project involves the creation of a system used in a university to aid students to enroll on various courses online, and aid department personnel to easily manage student and course data.

12 The Properties dialog can be also accessed by clicking on its icon on the TreeToolButtons toolbar, or by right-clicking on the package, and selecting Properties.

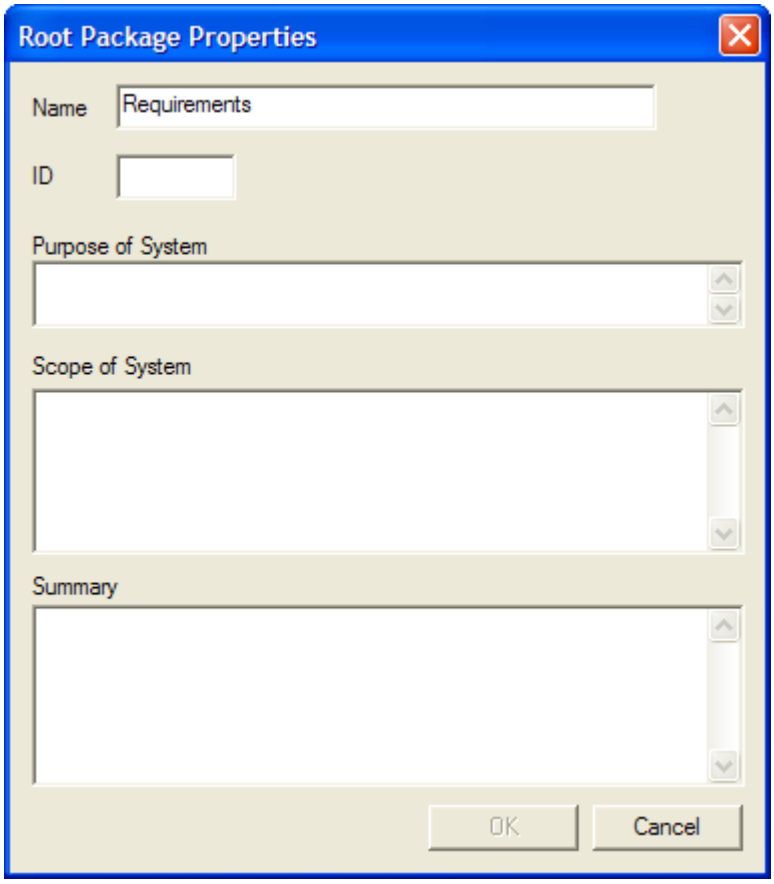

Image 7: Root Package Properties dialog

To create a new package as a child of the root package, right-click on it, and select New Package. The **Package Properties** dialog will appear. We will create a package named "Course Manipulation" (Image 8). Note that if we check the Auto ID check-box, *RaQuest* will assign IDs automatically to items. When we click on OK, the package will be added under the root package.

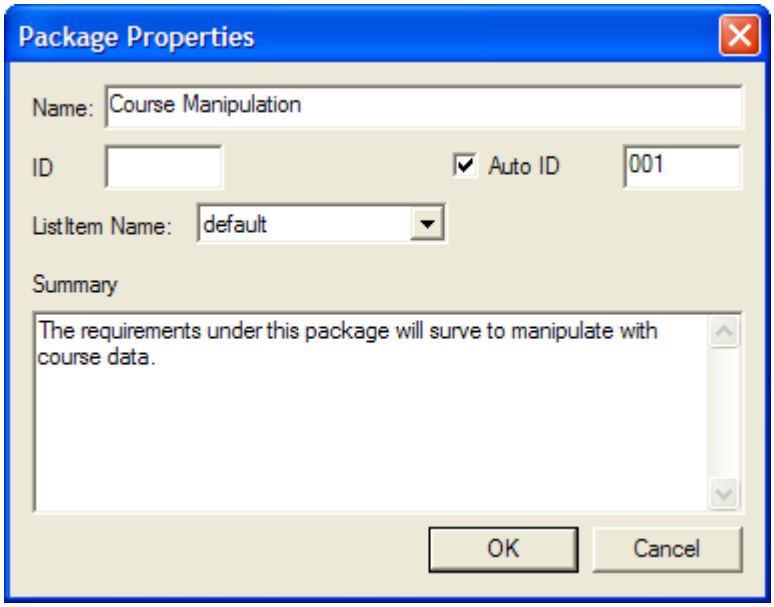

Image 8: The Package Properties dialog appears when we want to create a new package

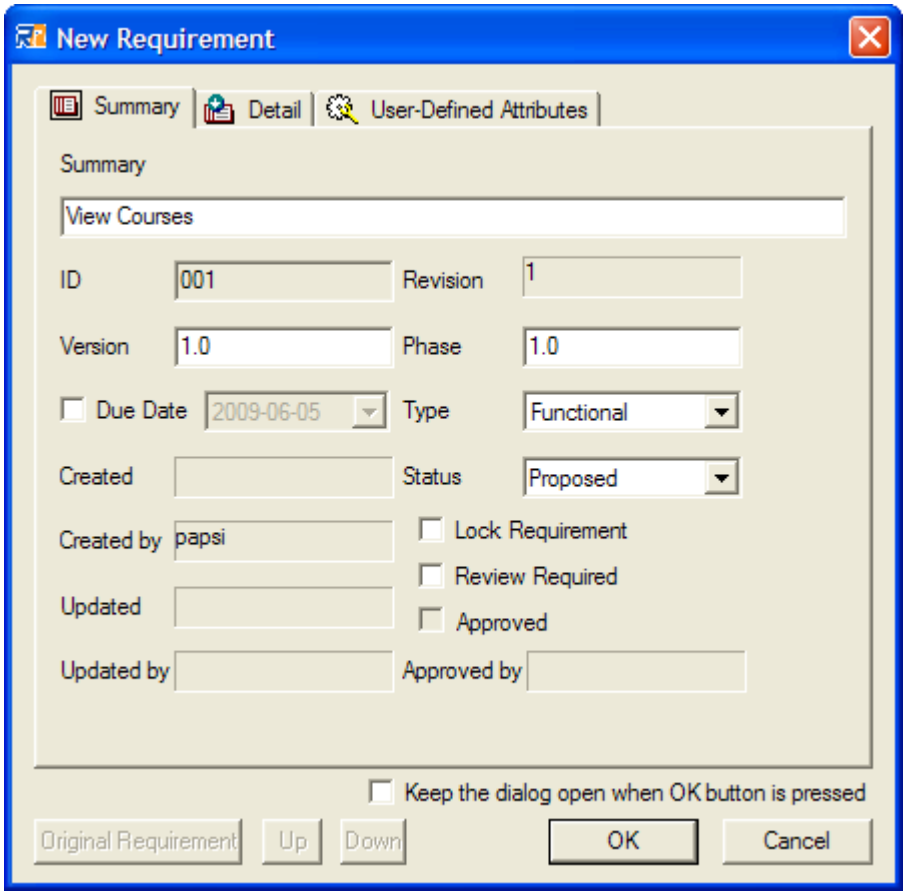

Image 9: The Summary tab of the New Requirement dialog

Next, we will add a new requirement by right-clicking on the previously created package, and selecting New Requirement. The **New Requirement** dialog will be shown. This dialog has three tabs:

- **Summary** (Image 9) – at this tab, we can give a name to the requirement, set its version and phase, select a due date (deadline), select its type and status. We can also lock the requirement, or force that a review will be required. The predefined types of a requirement are: "Functional", "Usability", "Reliability", "Performance", "Supportability", "Etc." and "-". The predefined statuses are: "Proposed", "Considering", "Validated", "Approved", "Duplicate", "Obsolete" and "Pending". Note, that a user can define his own requirement types and statuses in the Tools → Project Options dialog.
- **Detail** (Image 10) – we can enter here the textual description of the requirement and set other attributes like priority, difficulty, stability and risk[13] (all of these attributes can be "High", "Medium" or "Low"). We can also add additional information like effort needed or keywords. Note, that *RaQuest* will automatically calculate the numeric priority of the requirement based on all of these attributes and show the result in the Priority Value box.
- **User-Defined Attributes** – we can set here additional attributes if they are already defined in the Project Options dialog.

13 *RaQuest* defines the Risk attribute as the relative possibility that the requirement will cause future problems. The other attributes are defined traditionally.

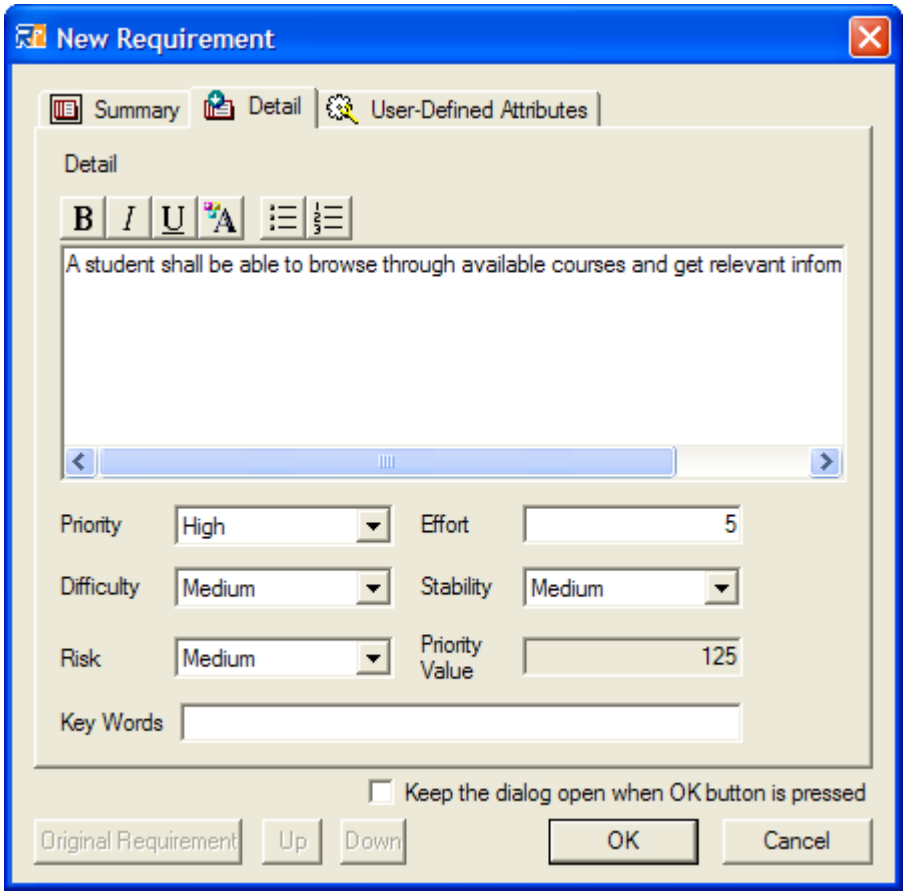

Image 10: The Detail tab of the New Requirement dialog

When we click on the OK button, the requirement will be added under the selected package, and it will be listed in the list pane (right pane). Image 11 shows the main window of RaQuest with three requirements.

We will now define some members to work on these requirements. In the main window, select the **Members** tab in the bottom of the tree pane (if it's not visible, select Tree → Member Tree). It is important to know, that in *RaQuest*, we can add divisions and members. **Members** are individuals and **divisions** are whole teams consisting of several members. We can define a division, and put members under that division as child elements. We can add and remove divisions and members just like requirements and packages. To add a new division or a member, right-click on the parent element, and select **Add Division** or **Add Member**. Members and divisions are simple elements (unlike requirements, only their name can be added or modified). Note, that a member can only be dragged and dropped to another division or to the root. Unlike requirements, a member cannot have another member as its child. Divisions can be dragged and dropped to another division (it will be shown as its child) or to the root. A sample window with an added division with six members can be seen on Image 12. We can assign members to requirements in three ways: by dragging the member from the member tree to a requirement in the list pane, by accessing the member matrix, or by opening the Members tab from the requirement Properties dialog. The Member Matrix (accessible from Matrix → Member Matrix) can show all assignments. On this matrix (Image 13), the green arrows represent assignments. We can also manage member assignments directly within this matrix. Simply select a table cell, right-click on it, and select the green arrow icon to assign that member to that requirement, or the blank icon to delete the assignment.

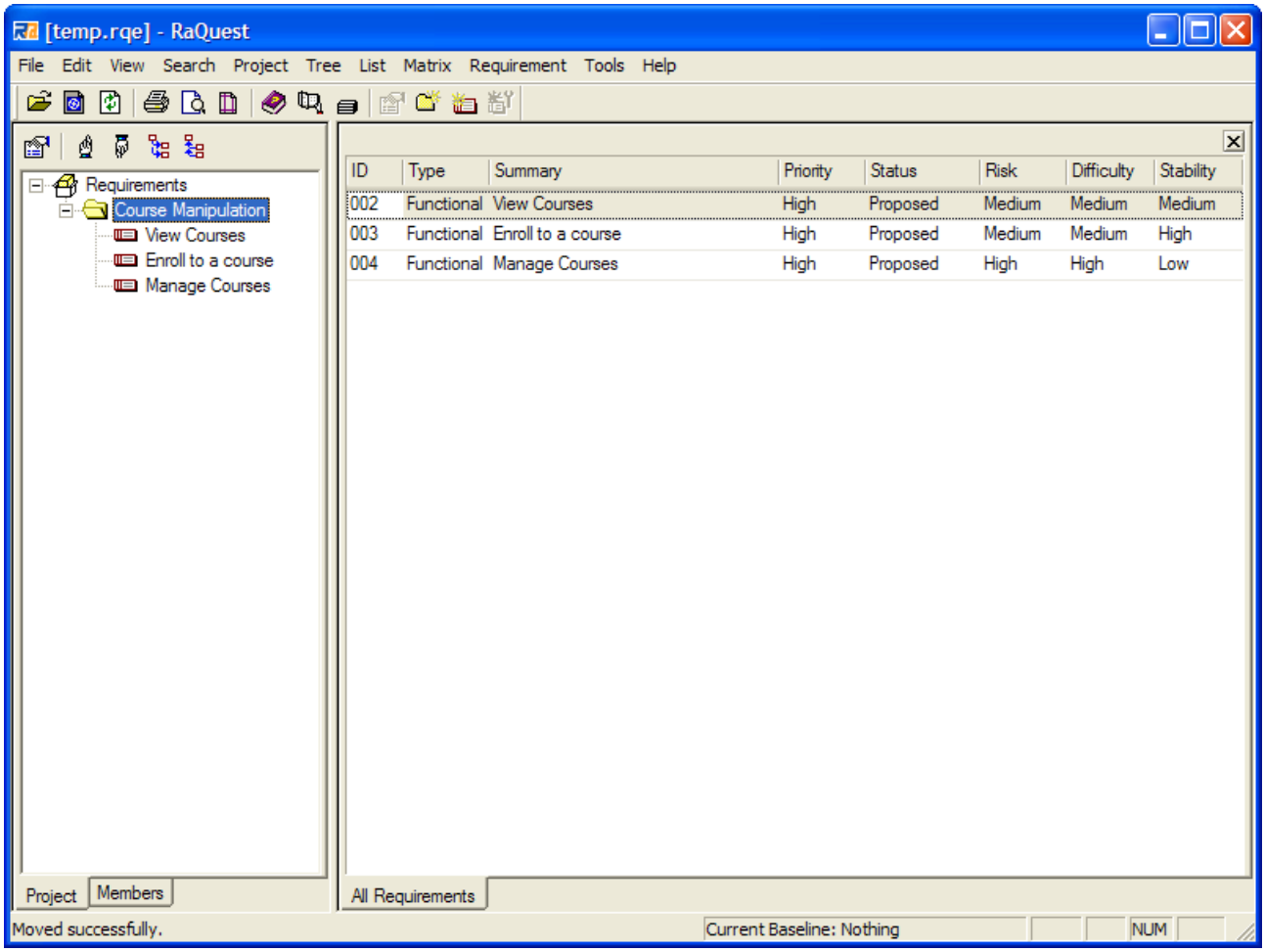

Image 11: Main window of RaQuest after three requirements has been added: "View Courses", "Enroll to a Course" and "Manage Courses"

Now, if we double-click on a requirement (from either panes), its **Properties** dialog will appear. This is the same as the New Requirement dialog, but with additional tabs:

- **UML Items** – enables us to link this requirement with UML items, like use cases.
- **Files** – enables us to link this requirement to external files (documents, models, etc.).
- **Members** (Image 14) – allows us to assign human resources to this requirement. To assign a member to a requirement, select the member from the drop-down list. We can also set the role of this member from the Role drop-down list. *RaQuest* has several predefined roles like "Assigned member", "Application analyst", "Business analyst", "Developer", "Project manager", "Solution architect", "Test architect" and "Use case modeler", but the user can also define new roles (the Roles drop-down list is editable). The following details are also available: time (time allocated to the member to complete the task), % Complete (progress of this task), start (start date of the assignment), finish (end date of the assignment) and desc. (textual description). To finally add the member to the requirement, click on the Add button. We can also assign a whole division to the requirement, if the Use Divisions as Member check-box is enabled in Tools → Project Options → General Features 1.
- **Test Items** (Image 15) – useful for the testing phase of the development. Here, we can define a test case by entering its name, and choosing its attributes: category (can be "Unit", "Integration", "System", "Acceptance" and "Scenario"), type ("Regression", "Standard",

"Load") and status ("Not run", "Pass", "Fail", "Deferred", "Canceled"). Also, we can assign this test to a member (by selecting the member from the Run By and Checked By drop-down lists). We can also add some textual messages, like description, input, acceptance criteria and results. Finally, when we click on the Add button, the test will be added.

- **Comments** – allows us to enter a comment to a requirement. Note, that multiple comments can be added, and *RaQuest* will show the whole list of comments together with the name of the commenter, and the date the comment was added.
- **Review Requirements** – displays the list of source requirements which cause the current requirement to be reviewed.
- **Update Log** – allows us to review the update log for the current requirement.

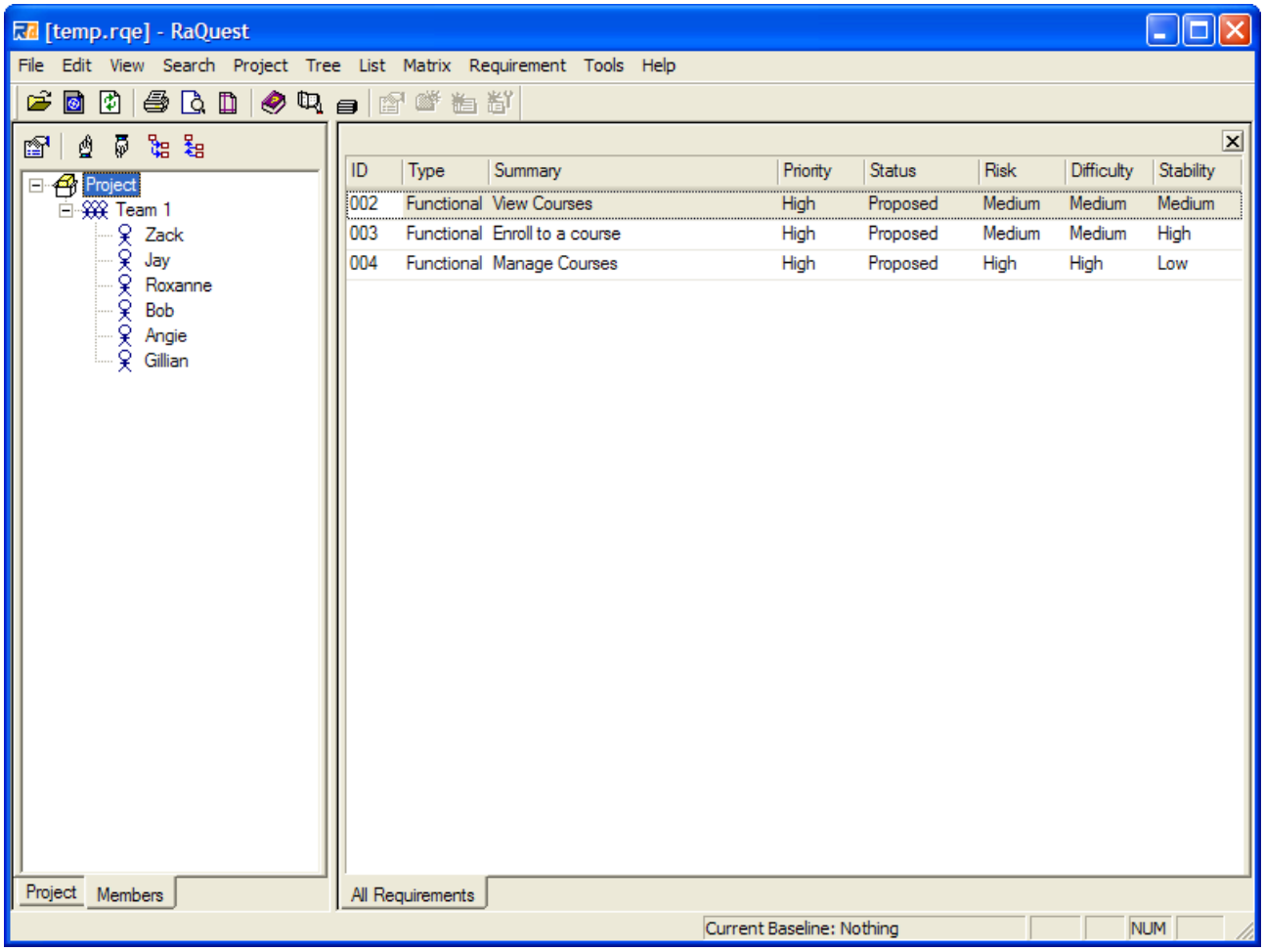

Image 12: A division of six members

As we know, *RaQuest* allows us to set the **status** of the requirements. The default status is always set to "Proposed", as this is the natural choice when a new requirement emerges. In this phase, a requirement can be freely modified. But approving a requirement is not so simple. Although there is an "Approved" status in the requirement Properties dialog, setting the status to "Approved" won't work (*RaQuest* simply won't allow it). This is because approving a requirement in reality isn't a simple task either. First, the requirement needs **validation**. When the requirement is reviewed, its status must be set to "Validated". This is a key step to approve a requirement in *RaQuest* (because only validated requirements can be approved). To validate a requirement in

RaQuest, double-click on that requirement, and in the Properties dialog set its status to "Validated". But as we click on the OK button, *RaQuest* will ask as the following question: "*If you change status to this, the requirement will be locked. Are you sure?*" If we choose Yes, *RaQuest* will lock this requirement by checking the Lock Requirement check-box. After this, all future modifications of this requirement will be rejected[14]. This is also a natural situation in requirements management, because it's not recommended to modify an already reviewed and validated requirement. When the requirement is finally validated, it can be approved. To approve a requirement in *RaQuest*, right-click on the requirement, and choose **Approval**. After a confirmation dialog, its status will be set to "Approved".

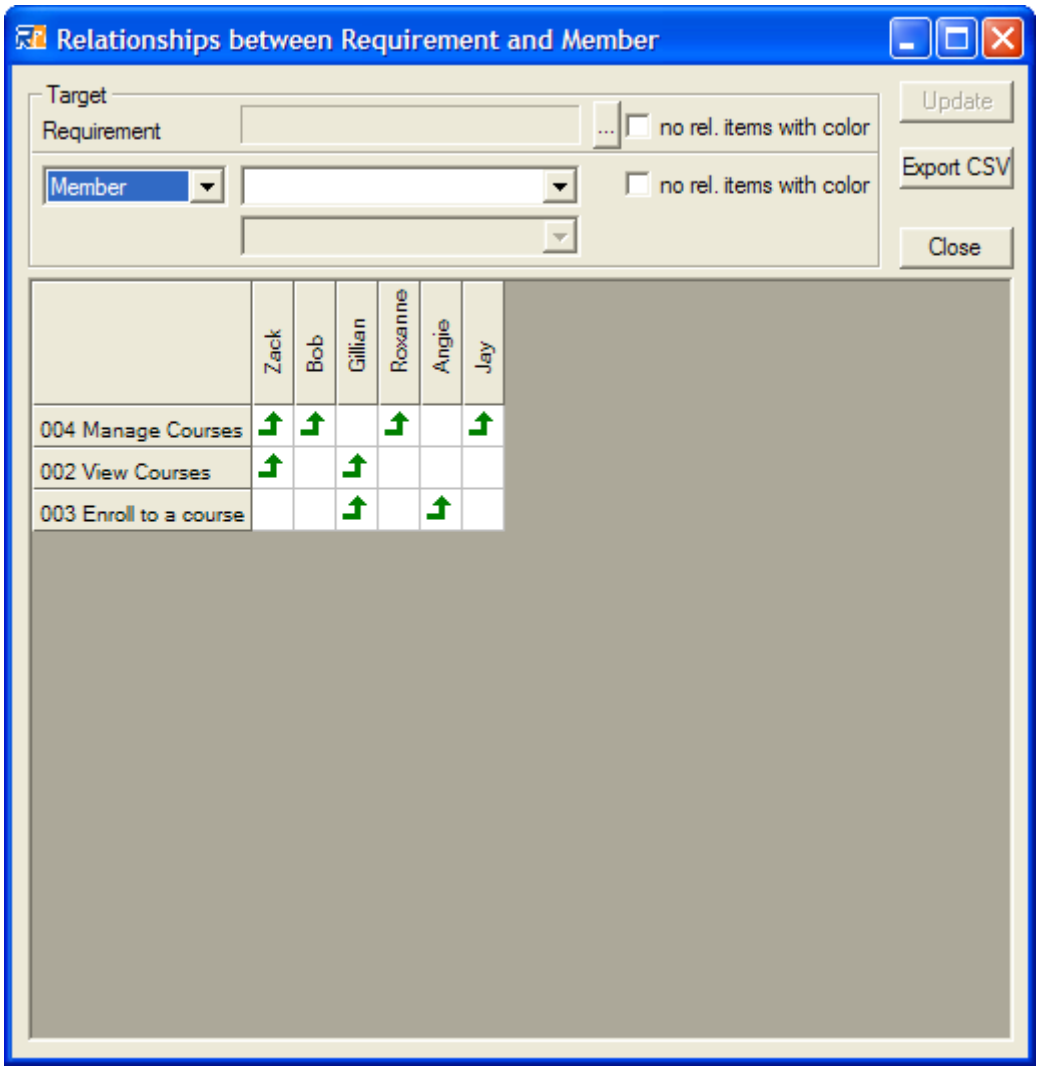

Image 13: The Member Matrix shows which members are assigned to which requirements (green arrow)

14 To be more precise, we can modify a requirement after it got validated or even approved, but only if we lift the lock by un-checking the Lock Requirement check-box in the requirement Properties dialog. *RaQuest* will warn us, that by doing this, the status of the requirement will be restored back to "Proposed".

Image 14: The Members tab of the requirement Properties dialog

We will now describe how to set **relationships** between requirements. Lets assume, that there is a relationship between the "View Courses" and "Manage Courses" requirements. In *RaQuest*, a relationship is defined by drag-and-drop methodology: simply drag the destination requirement from the tree pane to the source requirement in the list pane. Because "View Courses" depends on "Manage Courses", we will drag the "View Courses" requirement to "Manage Courses"[15]. On the status bar, the "*Relationship created successfully.*" message will be shown[16]. A relationship can be removed by right-clicking on the relationship and selecting Delete Relationship from the menu.

We can view relationships visually in several ways. First, we can try some **views** on the list pane (right pane). For example, we can select the package, and choose List → **Relationship List**. This will select all relationships of the selected package (Image 16). The "<" and ">" signs serve as arrows pointing from the destination requirement to the source. This list also allows us to delete relationships. Simply right-click on the relationship and select Delete Relationship. Of course, there are other views available, e.g. the **Destination Requirements** and **Source Requirements** lists. These views can show the list of destination and source requirements for the selected requirement, respectively. These views are accessible by right-clicking on the requirement, or from the List menu.

15 This means, that if we change the "Manage Courses" requirement, the "View Courses" requirement can also change.

16 Linking the requirements in both ways (linking requirement A to B, and then linking requirement B to A) is not possible in *RaQuest*. When we try to do that, the "*Failed to create relationship. Already assigned.*" message will be shown in the status bar.

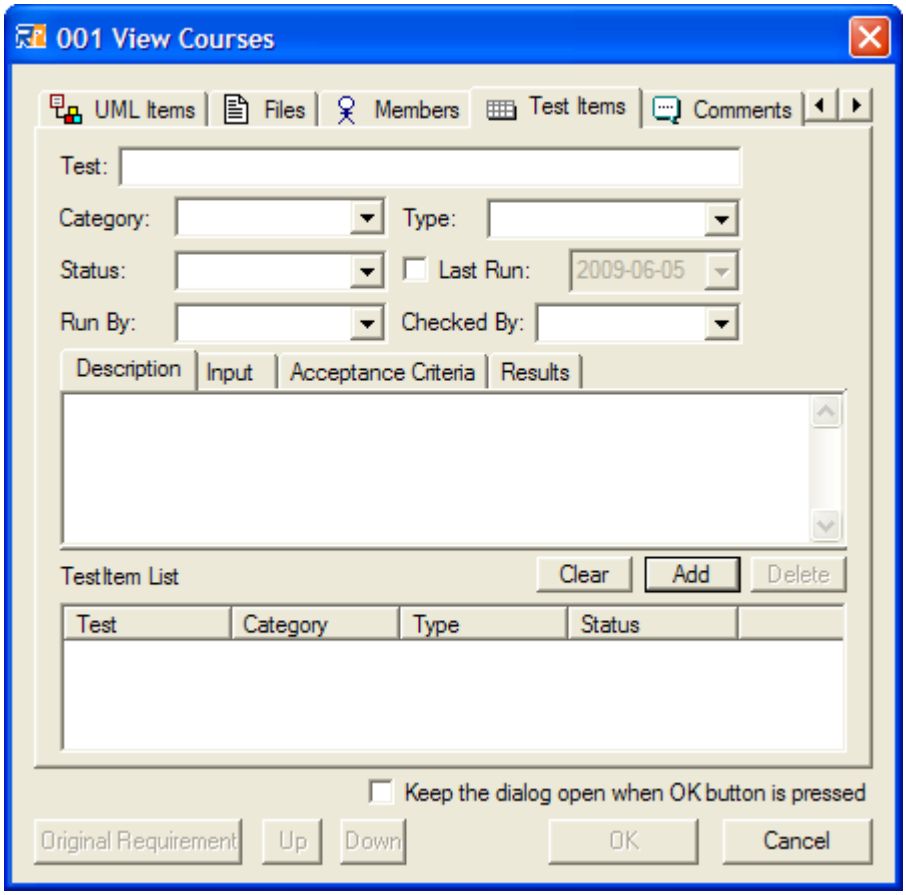

Image 15: The Test Items tab of the requirement Properties dialog

Another, more visual representation of relationships is the **Relationship Map**. We can access this map by right-clicking the observable requirement and selecting Show Relationship Map. Image 17 shows such a map for the "View Courses" requirement. The arrow here indicates the direction of dependency. As we can see, the observable requirement is colored red and the arrow points from the destination requirement to the source.

Finally, the most complete visual representation of relationships is the **traceability matrix**. In *RaQuest*, this is accessible by selecting Matrix → Requirements Matrix. In this matrix (Image 18) the relationships are represented with green arrows pointing from the destination requirement to the source. This matrix shows all requirements and relationships in one place. We can manage the relationships even here, because the table cells are interactive. To add, modify or delete a relationship, simply select a table cell, right-click on it, and select the appropriate arrow icon (or select the blank icon to delete that relationship).

In the next section, we will demonstrate some more advanced operations in *RaQuest*.

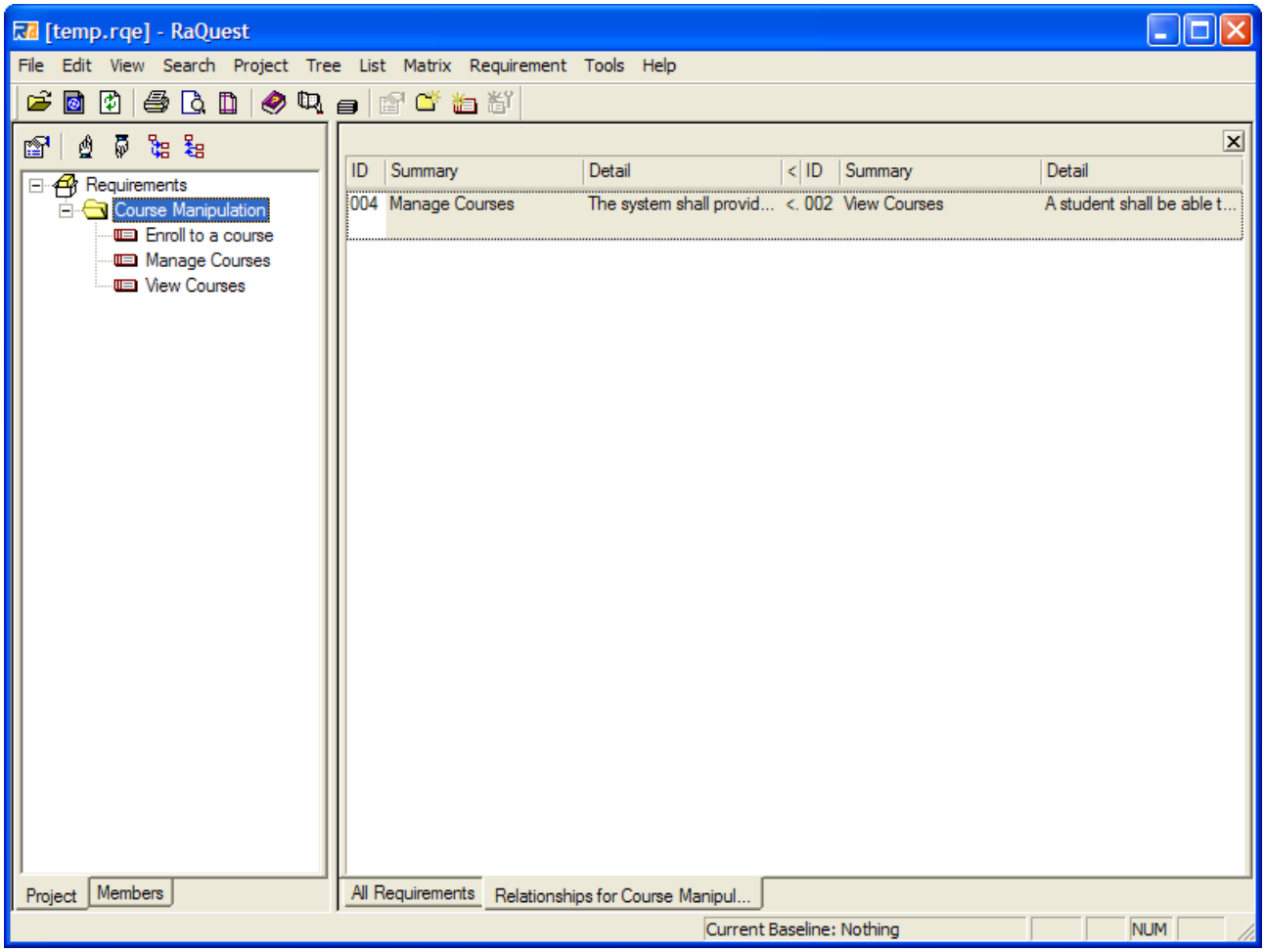

Image 16: Relationships of the "Course Manipulation" package

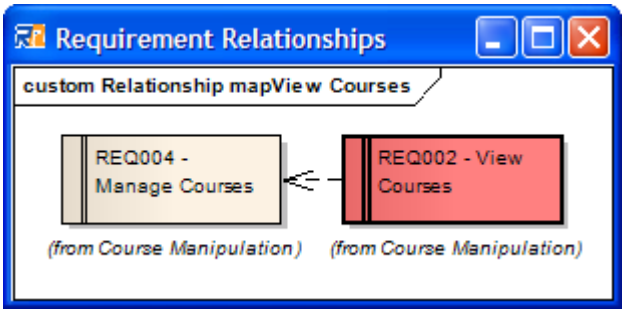

Image 17: Relationship Map of the "View Courses" requirement. It shows that it depends on the "Manage Courses" requirement. If "Manage Courses" changes, "View Courses" can also change.

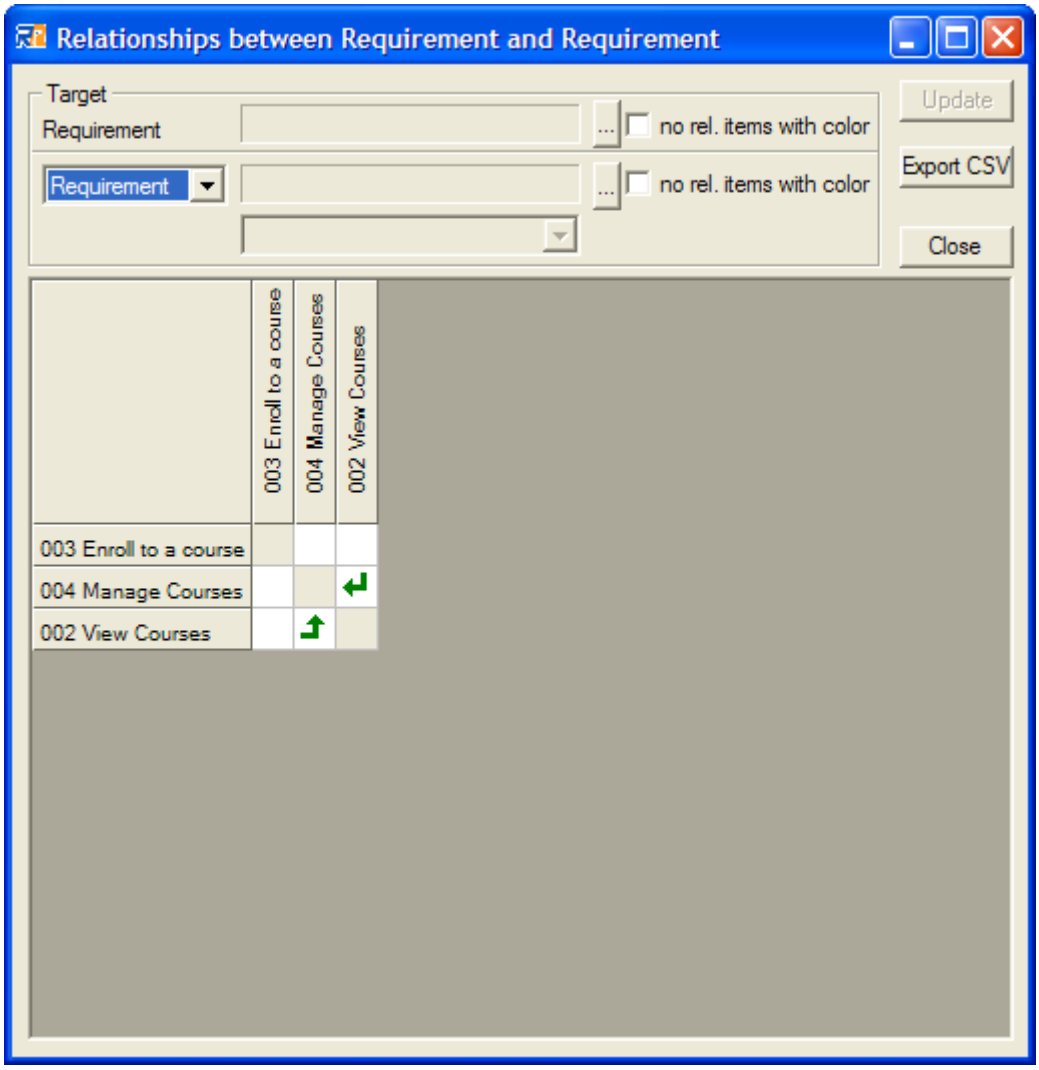

Image 18: The Requirement Matrix shows the relationships between all requirements

Some Advanced Operations

In this section, we will explain, how to integrate *RaQuest* with other applications like *Microsoft Word* or *Enterprise Architect*, how to search and filter requirements, and how to manage changes.

If we already have a vision document or a Software Requirements Specification (SRS) written in *Microsoft Word*, **importing** the requirements **directly from *Word*** to *RaQuest* would save us time. If we remember, during the installation of *RaQuest*, the setup program has also offered us to install the "*Word Addin*". This feature is disabled by default, which means that this is just a bonus feature in *RaQuest*, not a main one. We can also tell this by the possibilities this feature has. Although it has some basic automation, it's not fully automatic like some other requirements management tools.

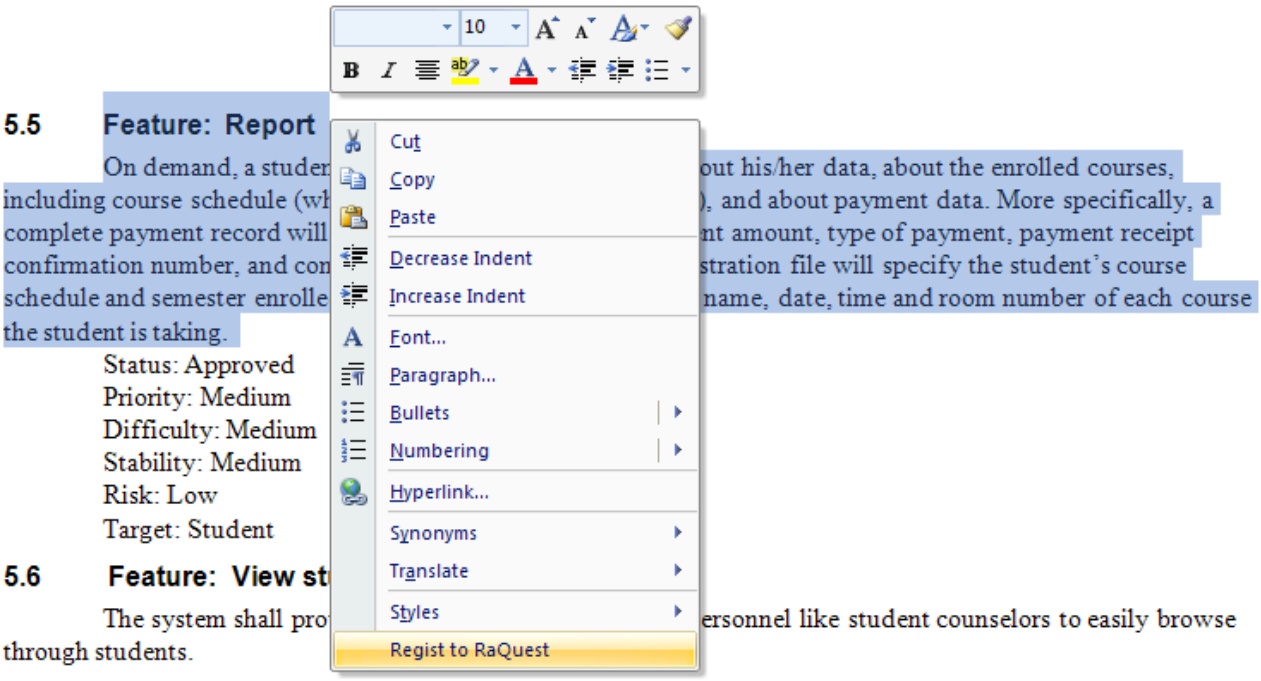

Image 19: Using the Word Addin feature in Microsoft Word 2007

To use the Word Addin feature, first we need to open a vision document or a SRS in *Microsoft Word*. The first limitation of this feature is that it cannot parse the document automatically by searching for specific keywords[17]. Instead, we must locate the requirements by ourselves, select the text, right-click on the selection, and choose Regist to RaQuest. If we select only one paragraph, it will be imported to *RaQuest* as the requirement's summary. If we select more than one paragraph, the first will be imported as the requirement's summary, and the other paragraphs as the requirement's detail. Image 19 shows a typical usage of this feature where two paragraphs were selected (one heading and one paragraph). When we click on Regist to RaQuest, a confirmation dialog will appear, and then the requirement will be ready in *RaQuest*. Note, that the newly registered requirement attributes will not be displayed immediately after registration. To solve this, the project must be reloaded (by choosing File → Reload).

The other (more important) application, with which *RaQuest* can integrate, is *Sparx Systems*' *Enterprise Architect*. *EA* can be used to import use case models to *RaQuest* and transform them into requirements. This integration is so seamless that the communication is practically bidirectional (even the project file extensions are the same). We will now demonstrate, how to **import use cases from *EA*** to *RaQuest*. We will use the sample Course Registration System seen before. Image 20 shows the use case diagram of the Course Registration System created in *EA*. First, we will open that *EA* project in *RaQuest*. This will load the project into *RaQuest*, but since no requirements were created in *EA*, both panes will be empty. To transform the use cases to requirements, select Tools → Enterprise Architect features → Import from this Project → Generate Requirements from Use Cases. In the Generate Requirements from Use Cases dialog we will select the use cases package, and finally by clicking on the Run button, the requirements will be ready in *RaQuest* (Image 21). Note, that the initial status of these requirements will be “Proposed”.

17 The other limitation is that it cannot recognize requirement attributes, so we must input them manually.

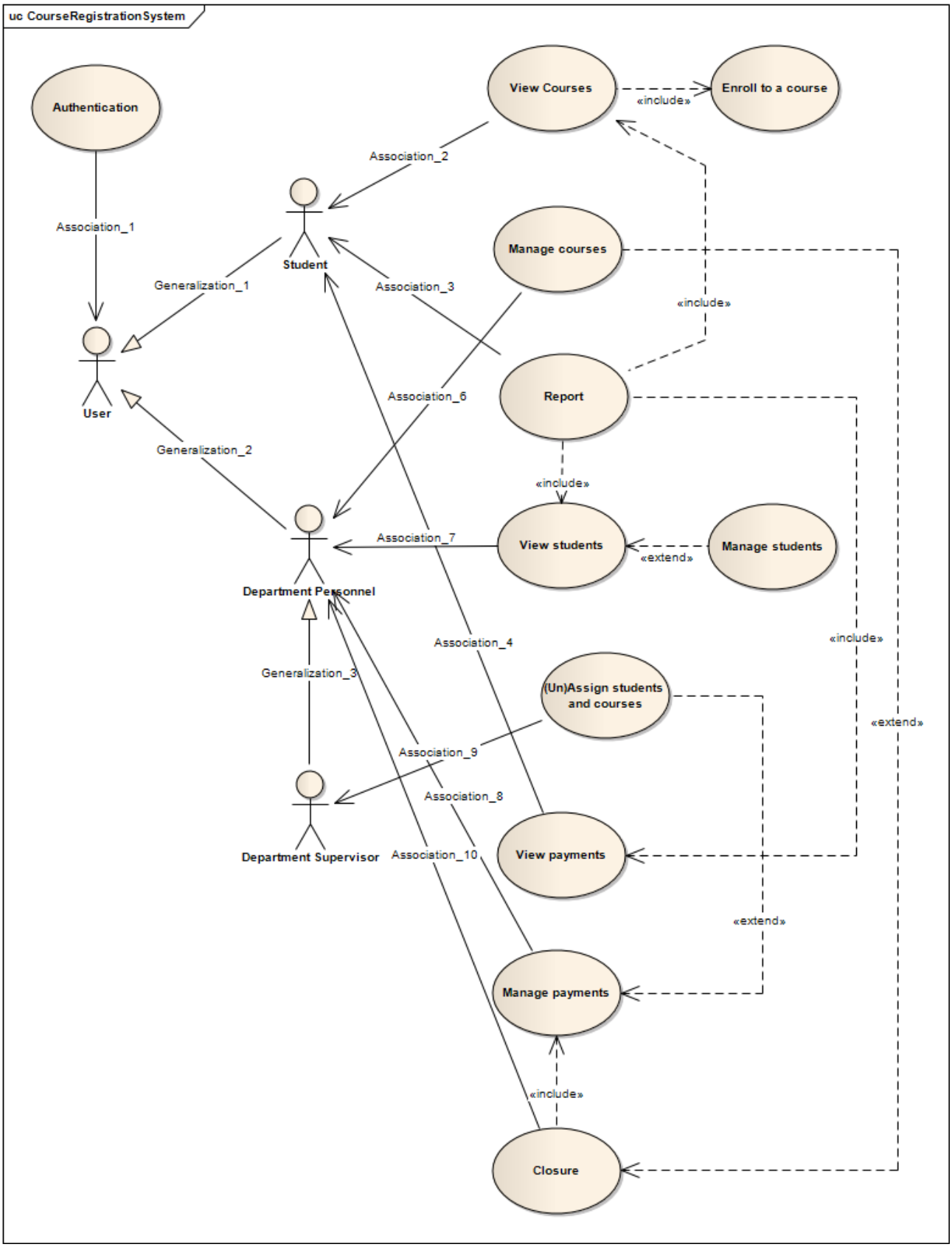

Image 20: The use case diagram of the Course Registration System as created in Enterprise Architect

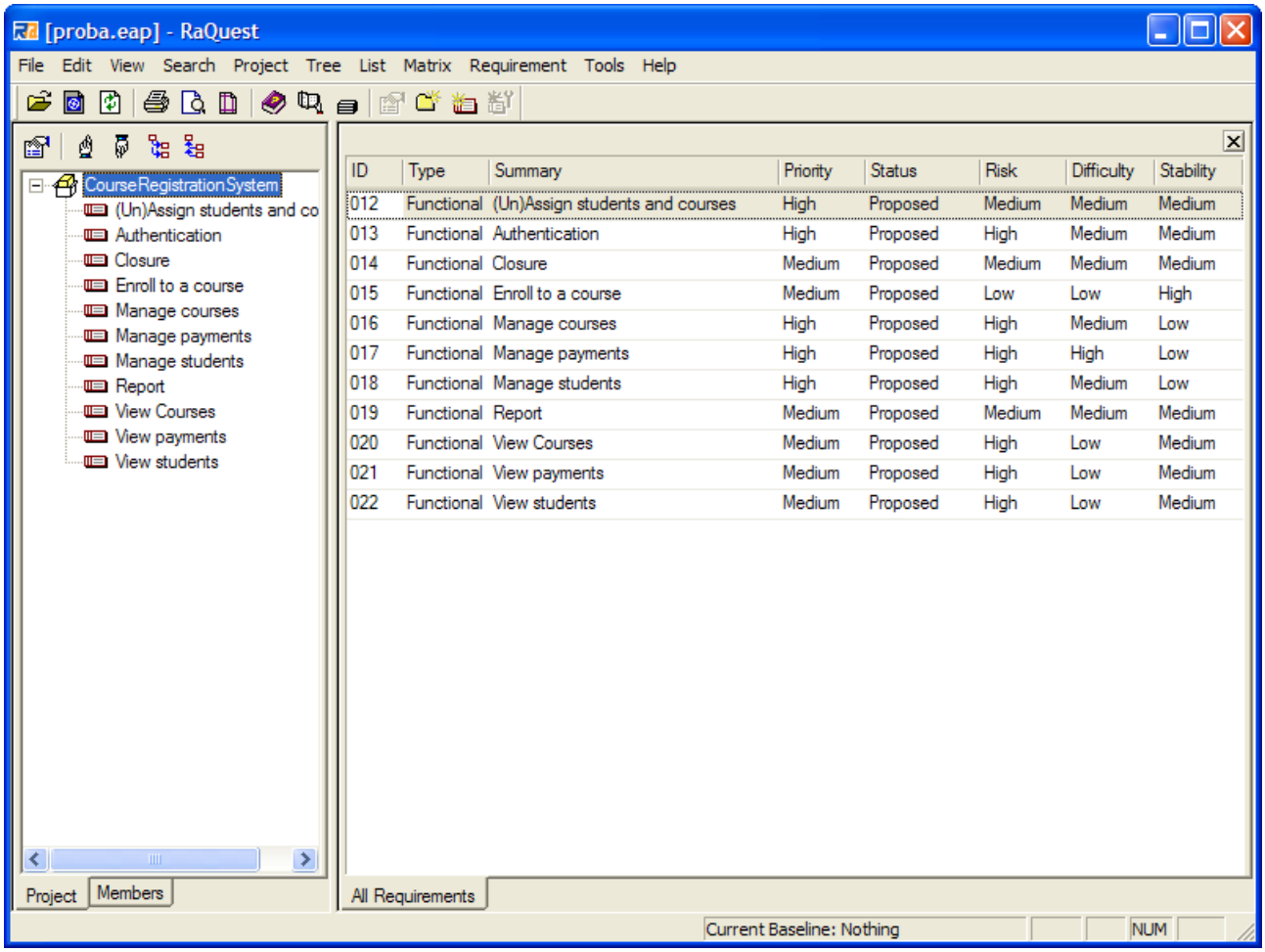

Image 21: The Course Registration System in RaQuest after the use cases were transformed to requirements

In reality, the number of requirements can be several hundreds or even more. In these situations, the search and filter commands can be extremely useful. To **search** for requirements, we must first select an initial search location from the list pane, and then choosing Search → Search. In the Search dialog (Image 22) we can search by requirement ID, summary, detail, type, status, search by members who updated or created the requirement, search by version, phase, priority, difficulty, user attributes, risk, stability, effort and date. By default, the search criteria are combined with the AND logical operator, but it can be replaced to the OR operator by choosing the or combo-box. Also, a negation can be defined by checking the Exclude check-box. By default, the search is case sensitive, but it can be disabled by checking the Ignore Case check-box. Finally, when we click on the Run button, *RaQuest* will begin the search by checking all requirements in the list pane from the initial location **downwards**. To search **upwards**, select the Up combo-box in the Search dialog. When *RaQuest* founds a match, it will select that requirement, and it will display the "*Found.*" message in the status bar. To continue with the search, select Search → Search Next, or simply press F3 on the keyboard. When *RaQuest* cannot find a match, it will display the "*Not Found.*" message in the status bar.

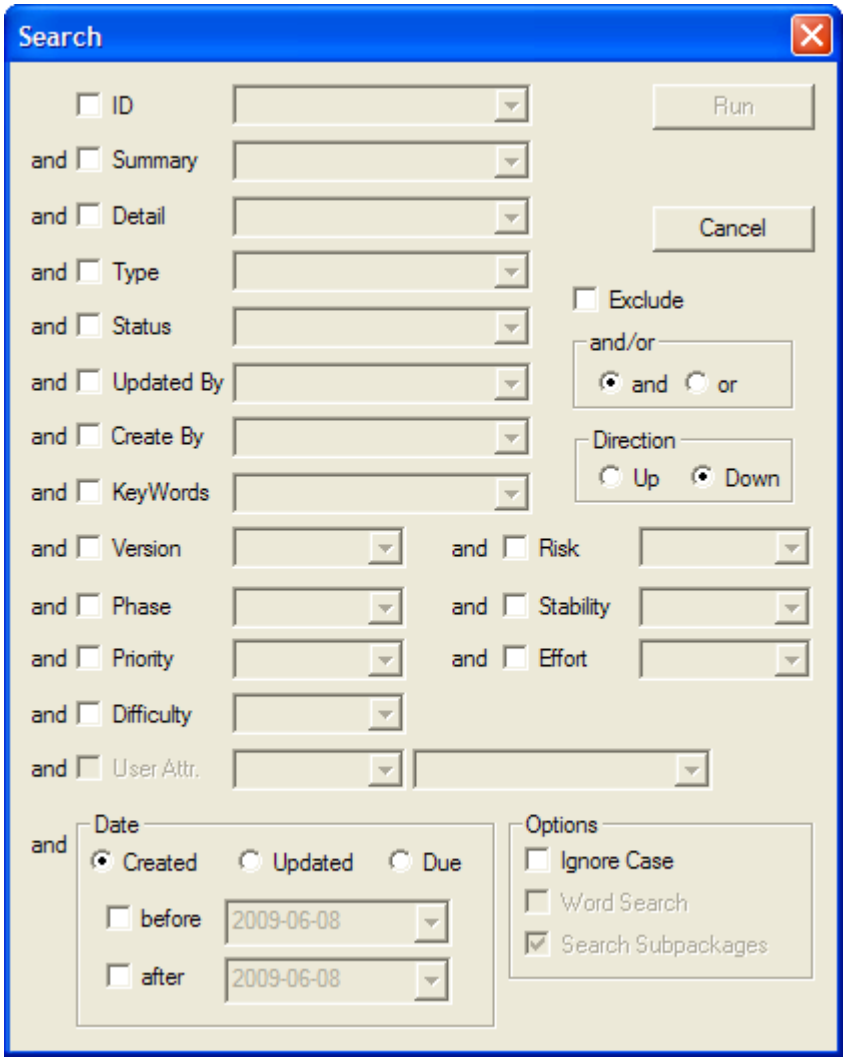

Image 22: The Search dialog

The Search command allows the user to search for a specific requirement. If we want to find all requirements which satisfies a given condition, we will **filter** the list. This command returns a **subset** of requirements satisfying the given condition. Note, that *RaQuest* will filter the currently active (selected) list pane (for example, if currently the All Requirements list pane is active, then *RaQuest* will treat this list as the initial set of requirements). To filter the list of requirements, select the initial list, and choose Search → Filter List. We will notice that the Filter List dialog is almost the same as the Search dialog. Only two differences can be seen. First, the Direction combo-box is omitted for obvious reasons. Second, a new button is available: (New Tab). This button is the same as the Run button, but it will display the resulting list in a completely new tab (the Run button will rewrite the contents of the initially selected tab). For example, if we want to list all requirements with ID 014 and all requirements which contains the "view" string in the summary text-field (case insensitively), then we will filter as shown at Image 23. if we click on the (New Tab) button, the resulting list will be displayed in a new tab (Image 24).

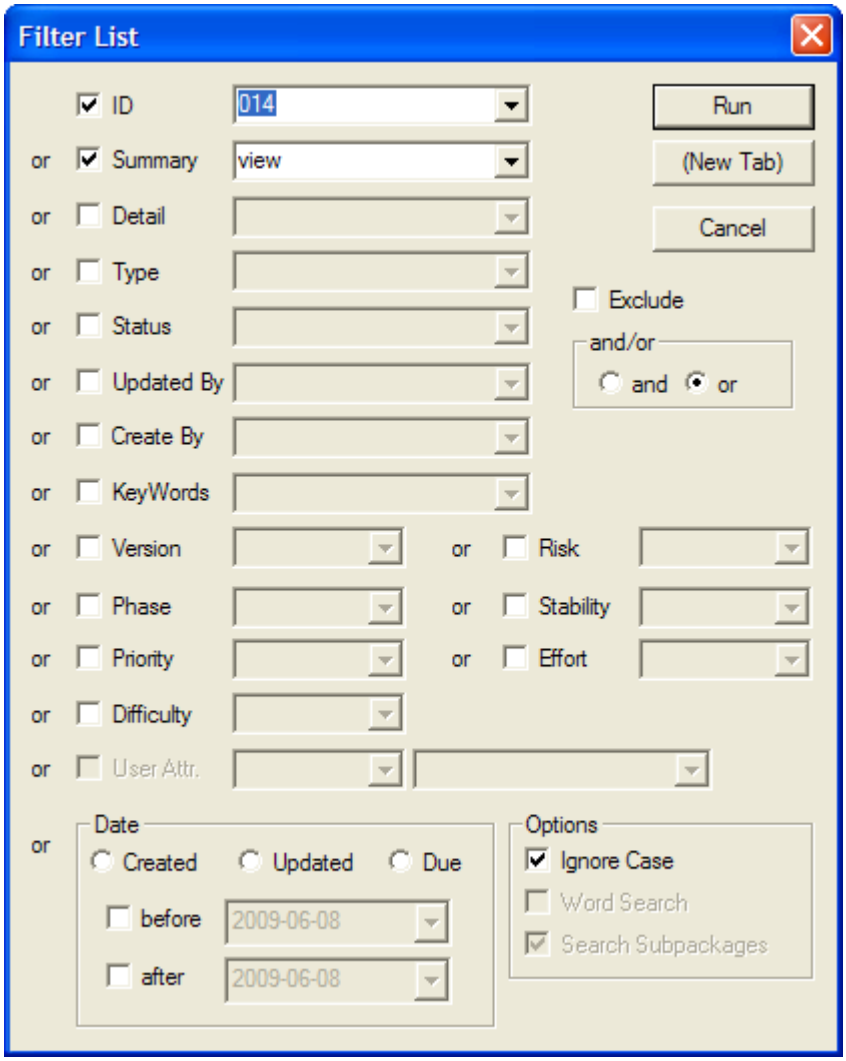

Image 23: The Filter List dialog. In this case, we want RaQuest to return all requirements with ID=014 and with the string "view".

We will now explain, how to define a **baseline**. As we have discussed previously, *RaQuest* treats the word “baseline” a little differently (see page 25). Lets assume that we want to save a snapshot of the current status of the project. To save a **snapshot**, go to Project → Manage Baselines. Because we haven't defined a baseline yet, the Manage Baselines dialog will be empty (Image 25). When we click on the New Baseline button, the Baseline dialog will appear, where we can define the name and version of the project snapshot. When we click the OK button, the snapshot will be created. For now, we can close the Manage Baselines dialog.

We will demonstrate the possibilities of the baseline feature by modifying and adding a requirement and by defining a change to an existing requirement. Lets assume, that after some reconsideration, the team has decided to **modify** the “Report” requirement. As it has turned out, the stability of this requirement is supposed to be low, because it calls many other functions. Also, the risk is supposed to be high for the same reason. No other modifications were suggested. The team has modified this requirement by **lifting** the lock (in the Properties dialog, un-check the Lock Requirement check-box), but because *RaQuest* has changed the status of the requirement back to “Proposed”, the requirement has needed to be validated and approved again.

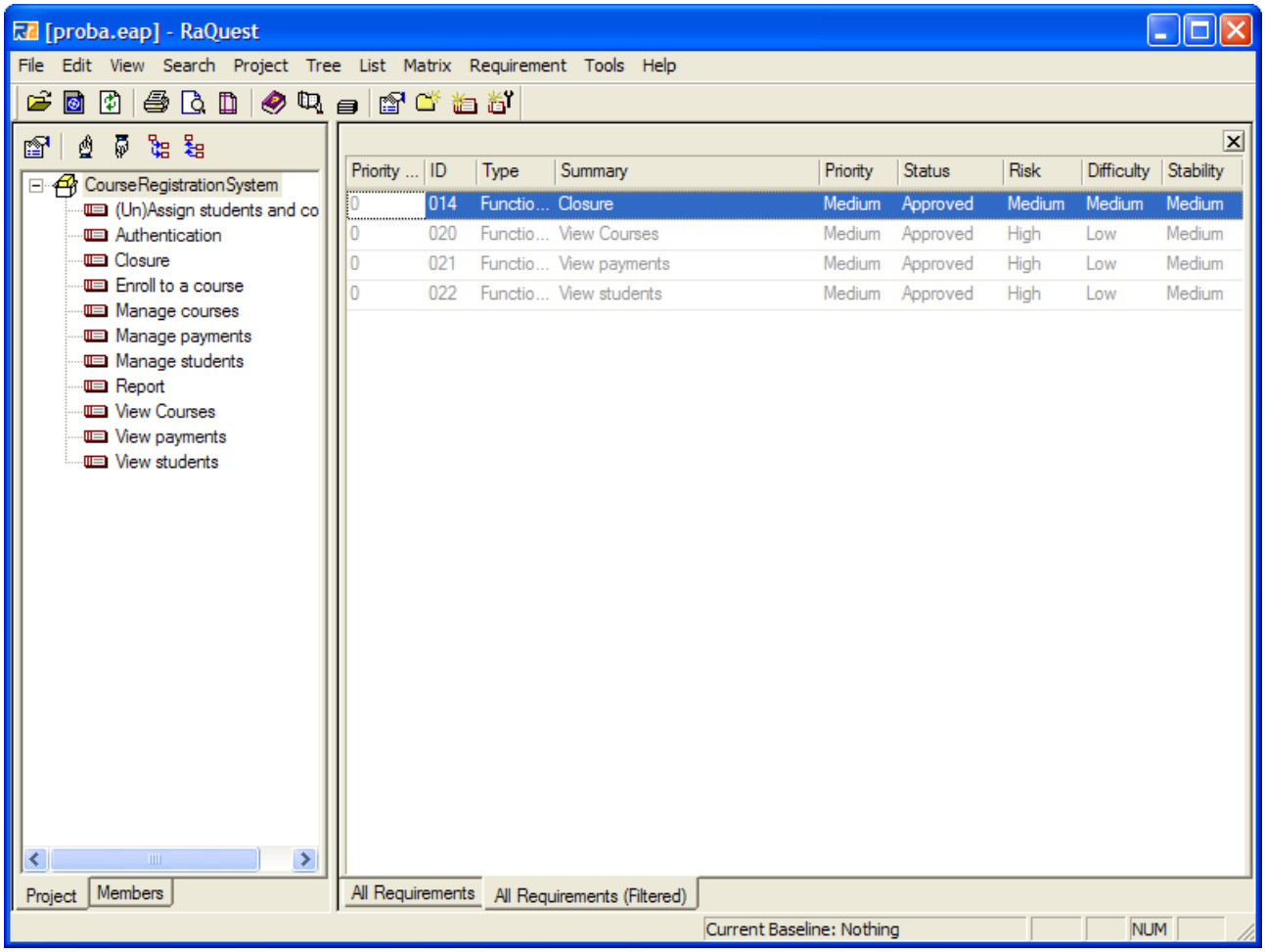

Image 24: The resulting subset of requirements shown in a new tab (notice the suffix "Filtered")

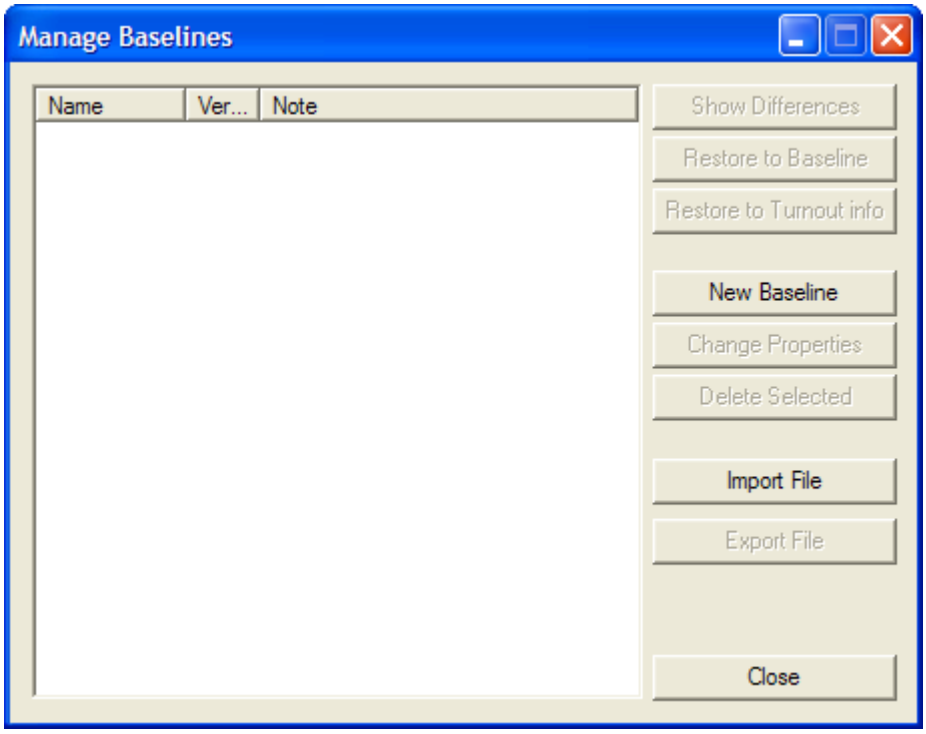

Image 25: The Manage Baselines dialog

Lets assume now, that in a later stage of the project, the project stakeholders have decided to also store professor data in the system. This would mean additional tasks for the developers, like creating new databases for the professors and managing this data. The team has decided to define a **new requirement** to satisfy this new demand: "Manage Professors". After the creation of this requirement, the team has defined its traceability, and it has turned out that it would be useful for the "Report" requirement to also include professor data in the student report. With other words, it was decided to change that requirement.

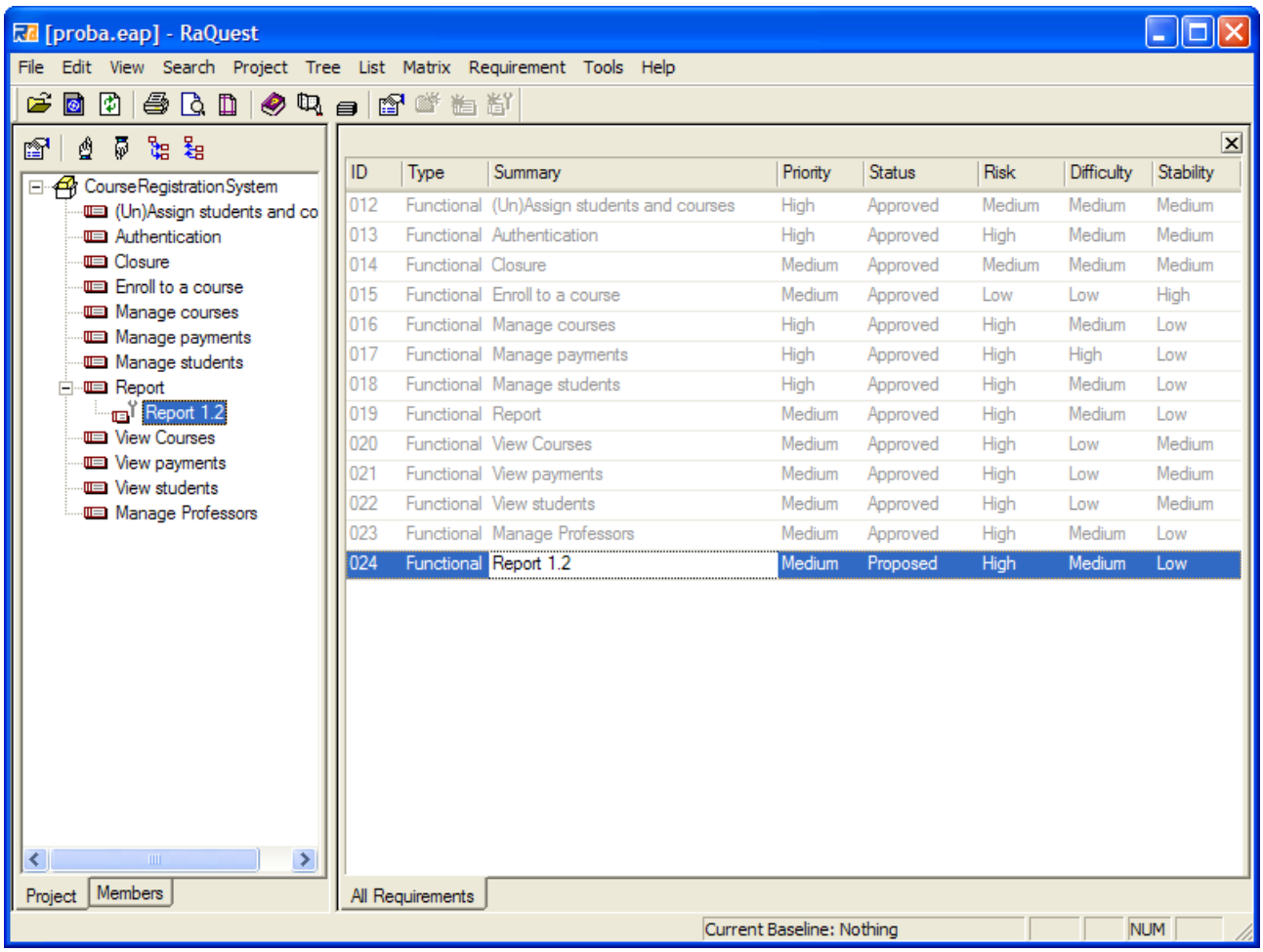

Image 26: A new change to a requirement is defined as a child requirement. The icon of the child requirement is also changed.

As we know, *RaQuest* allows us to define changes to requirements in a very elegant way. To define a **new change** to a requirement, right-click on that requirement, and select New Change[18]. The New Change dialog appears, which is exactly the same as the New Requirement dialog. The only difference is a new button (Original Requirement), which will show the old requirement that will be changed. When we click on the OK button, the change requirement will be made as a child of the old requirement (Image 26), and the icon will be replaced to indicate that this is a change (and not a simple parent-child relationship). Also, a relationship will be created automatically between these two requirements.

18 Note, that only an approved requirement can be changed.

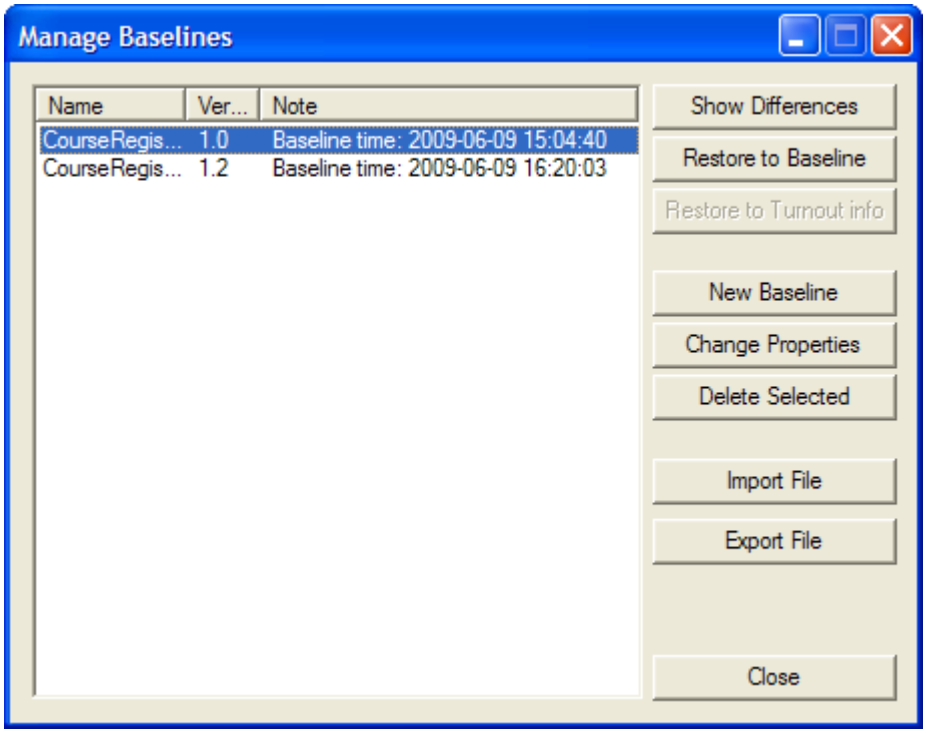

Image 27: The Manage Baselines dialog with two saved snapshots

Because this will be a **new version** of the system, it is recommended to save a new snapshot of the project status. To do this, we will go back to the Manage Baselines dialog (Project → Manage Baselines), and click on the New Baseline button. After the new snapshot was saved (Image 27), we can select the old version, and click on the Show Differences button to show what was changed or modified between the old and the current status of the project. We can even export a snapshot to an *XML* file by clicking on the Export File button.

We will conclude this chapter by showing the final traceability matrix of the requirements of the Course Registration System (Image 28).

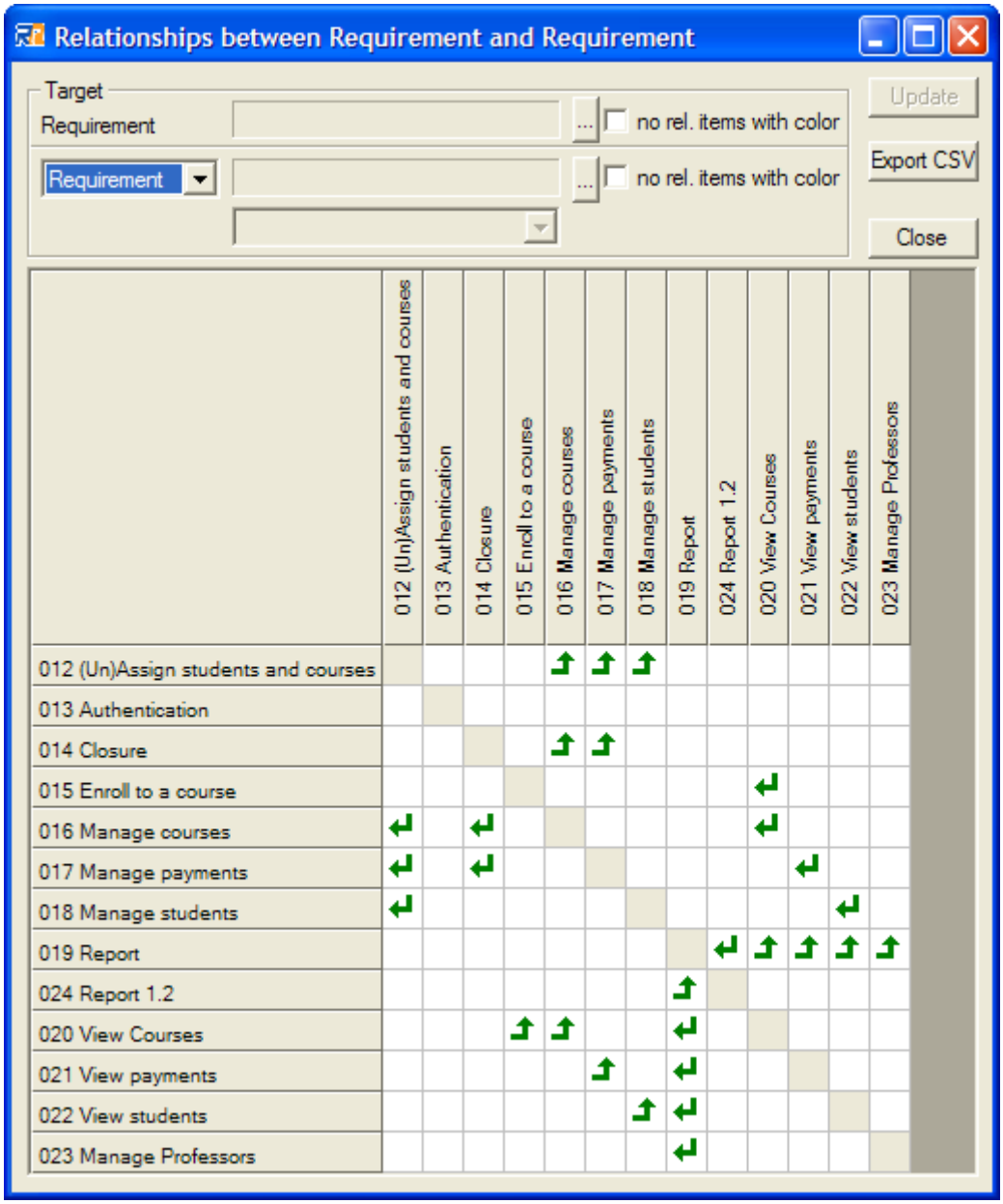

Image 28: A more complex Requirement Matrix shows all relationships between requirements in the Course Registration System

Conclusion

In this term paper we have discussed first a bit more generally about requirements management and requirements management tools. In the main chapter, a specific requirements management tool was presented: *RaQuest v3.1* from *SparxSystems Japan*. This tool is interesting because we cannot really classify it neither as a document-centric, nor a database-centric tool. Instead, it integrates itself with a UML tool named *Enterprise Architect* from *Sparx Systems*. Although it has the ability to work together with *Microsoft Word*, we will get much better results if we integrate it with use case diagrams and requirements made in *Enterprise Architect*. If we can import a use case diagram made in another UML modeling tool into *Enterprise Architect*, we can import it into *RaQuest*, too.

After the short introduction of *RaQuest*, we have described how this tool works. As a case study, a sample Course Registration System was used. The main advantages of *RaQuest* are its seamless integration with the UML tool *Enterprise Architect*, lightweight behavior of the tool itself (the installation file is only 6 megabytes, and it's very resource-friendly during usage), solid number of features and its cheap price (compared to other tools). But *RaQuest* has its own weaknesses, too. For example, it requires *Enterprise Architect* to run, so if the company doesn't have this UML tool, licensing will be more expensive (because the company will also need to buy *Enterprise Architect*). Fortunately this is not a big problem, because *Enterprise Architect* is very popular nowadays. Another weakness of *RaQuest* is that the possibilities of integration with *Microsoft Word* are limited (no automatic parsing, requirement attributes can't be recognized). The last problem with this product is that training is only available in Japan. Fortunately, *RaQuest* has an easy-to-use user interface with informative help manuals, so mastering *RaQuest* is definitely possible.

References

[Wiegers, 2003]: Karl E. Wiegers 2003, Software Requirements, Second Edition, Microsoft Press, Redmond

[Paulk et al., 1994]: Mark C. Paulk, Charles V. Weber, Bill Curtis & Mary Beth Chrissis 1994, The Capability Maturity Model: Guidelines for Improving the Software Process, Addison-Wesley, Boston

[Shore & Warden, 2008]: James Shore & Shane Warden 2008, The Art of Agile Development, O'Reilly, Sebastopol

[Jarke, 1998]: Matthias Jarke 1998, 'Requirements Tracing', Communications of the ACM, vol. 41-12, pp. 32-36

[SparxSystems RaQuest 3.1 Feature Guide, 2009]: Requirement Management Tool RaQuest 3.1 Feature Guide 2009, Techn. documentation, SparxSystems Japan, viewed 28. May 2009, <http://www.raquest.com/downloads/RaQuestGuideEn.pdf>

[SparxSystems RaQuest 3.1 Install Manual, 2009]: RaQuest 3.1 English Version Install Manual 2009, Techn. documentation, SparxSystems Japan, viewed 28. May 2009, <http://www.raquest.com/downloads/RaQuestInstallEn.pdf>

[SparxSystems RaQuest 3.1 Startup Manual, 2009]: RaQuest 3.1 Startup Manual 2009, Techn. documentation, SparxSystems Japan, viewed 28. May 2009, <http://www.raquest.com/downloads/RaQuestStartupManualEn.pdf>

[SparxSystems RaQuest Help, 2009]: RaQuest Help 2009, Techn. documentation, SparxSystems Japan, viewed 28. May 2009, <http://www.raquest.com/downloads/RaQuest.chm>

[Sparx Systems, 2009]: Sparx Systems 2009, , viewed 28. May 2009, <http://www.sparxsystems.com.au/>

[SparxSystems Japan Co., 2009]: SparxSystems Japan Co. 2009, , viewed 1. June 2009, <http://www.raquest.com/>

[Laatukonsultointi P. Kantelinen Oy, 2009]: Laatukonsultointi P. Kantelinen Oy 2009, Requirements Management Tools, viewed 28. May 2009, <http://www.laatuk.com/tools/reg_mgmt_tools.html>

[International Council on Systems Engineering, 2009]: International Council on Systems Engineering 2009, About INCOSE, viewed 28. May 2009, <http://www.incose.org/about/index.aspx>

[INCOSE Tools Database Working Group, 2009]: INCOSE Tools Database Working Group 2009, INCOSE Requirements Management Tools Survey, viewed 28. May 2009, <http://www.paper-review.com/tools/rms/read.php>