9 DOORS: A Tool to Manage Requirements

There's nothing remarkable about it.
All one has to do is hit the right keys at the right time
and the instrument plays itself.

Johann Sebastian Bach, composer, 1685–1750

9.1 Introduction

Systems engineers and managers need the right instruments to assist them with the requirements management process. A variety of tools currently exist. This chapter presents an overview of one of these tools – DOORS (Version 7.1). DOORS (Dynamic Object Oriented Requirements System) is a leading requirements management tool used by tens of thousands of engineers around the world. The tool was originally created by QSS Ltd, Oxford, and is now developed and marketed by Telelogic.

DOORS is a multi-platform, enterprise-wide requirements management tool designed to capture, link, trace, analyze and manage a wide range of information to ensure a project's compliance to specified requirements and standards. DOORS provides for the communication of business needs, allows cross-functional teams to collaborate on development projects to meet these needs and enables one to validate that the right system is being built, and is being built right. The views provided by DOORS on the screen provide a powerful familiar navigation mechanism.

Throughout this chapter reference will be made to a case study for a family sailing boat.

9.2 The Case for Requirements Management

Today, systems engineers require effective requirements management in order to provide solutions. Requirements management is the process that captures, traces and manages stakeholder needs and the changes that occur throughout a project's lifecycle. Products, too, are becoming more complex, to the point where no individual has the ability to comprehend the whole, or understand all of its constituent parts. Structuring is by far the best way of organizing requirements, thus making them more manageable in terms of omissions or duplicate information. Hence requirements management is also about communication. For that reason,

it is important that requirements are communicated correctly, thus ensuring that team collaboration is enhanced, project risk is reduced and the project meets its business objectives. If requirements are well managed, the right product will get to market on time, on budget and to specification.

9.3 DOORS Architecture

For any application, the requirements and related information can be stored in a central database in DOORS. This database can be accessed in a variety of ways and exists throughout the lifetime of the application. The information in a DOORS database is stored in modules (Figure 9.1). Modules can be organized within the database by using folders and projects. A project is a special kind of folder that contains all the data for a particular project.

DOORS *folders* are used to organize data and are just like folders in a computer file store. Folders may contain other folders, projects or modules. Folders are given a name and description and the ability for users to see or manipulate the data in a folder may be constrained using access controls.

DOORS *projects* are used by a team of people to manage a collection of data relating to that team's work effort. The project should contain all of the data related to the requirements, design, development, test, production and maintenance for an application. The project provides the capability to manage users and their access to the data in the project, to back up the data and to distribute portions of the data to other DOORS databases.

DOORS *modules* are containers for data sets. Three classes of module exist:

- *formal* – the most frequently used type of module for structured sets of like information;

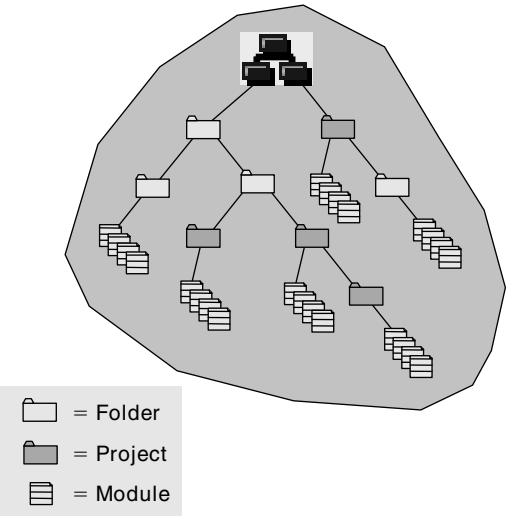

Figure 9.1 DOORS database structure.

- *descriptive* – unstructured source information (letters or interview notes);
- *link* – contain relationships between other data elements.

The user interface works very much like Windows Explorer and lets the user navigate through the database.

9.4 Projects, Modules and Objects

9.4.1 DOORS Database Window

The DOORS Database window allows the user to see and manage the organization of DOORS data. Figure 9.2 shows the database window, with the database explorer to the left and the list of contents of the selected folder on the right.

DOORS provides the capability to change the name or description of existing folders and projects. Folders and projects can also be moved if there is a need to reorganize or change the structure of the database. Folders and projects can also be cut, copied or pasted within the database to reorganize or duplicate portions of the database.

9.4.2 Formal Modules

Using the DOORS Database window, a new formal module can be created using the menu **File ▶ New ▶ Formal Module** as shown in Figure 9.3. The name of

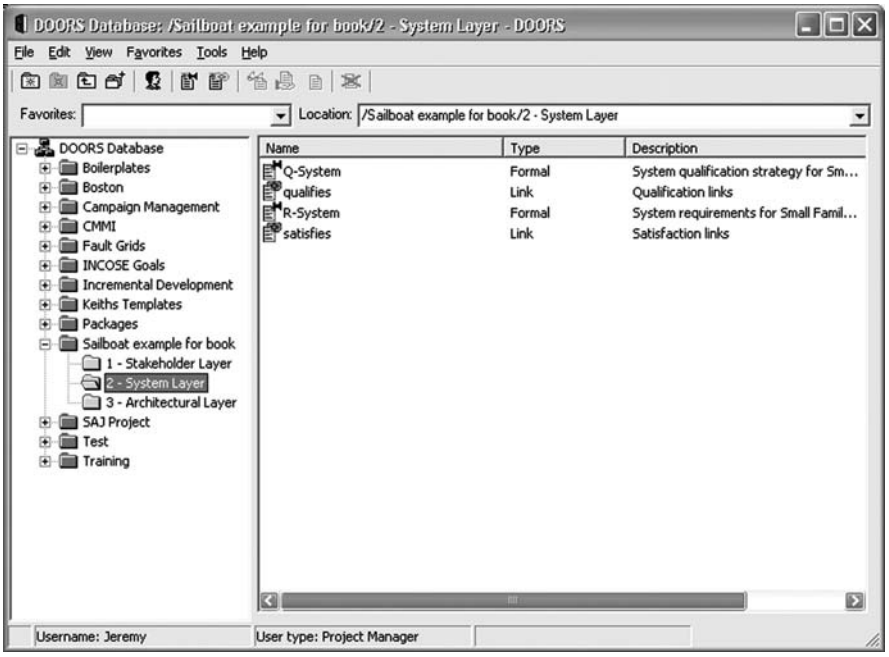

Figure 9.2 Database window.

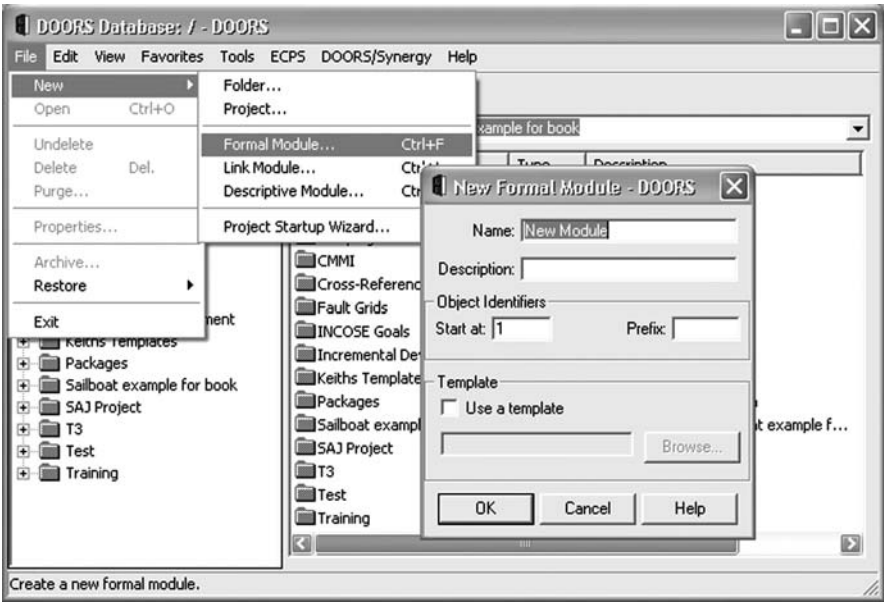

Figure 9.3 Create new formal module.

the new module and its description can be input. The unique identifiers generated for the objects in the module are numbered using the legal numbering system.

A prefix can be added to this number that reflects the contents of the module, such as PR for product requirements. By defining a unique prefix for each module, a project-wide unique identifier for all information in the DOORS project is established. This provides a convenient reference.

When a formal module is opened, the default display shows the module explorer on the left, and the module data on the right as shown in Figure 9.4.

The module explorer makes it easy to move to a specific place in the document and also shows the structure of the information in the module. Sections can be expanded or collapsed in the same way as can be done with Windows Explorer.

The right-hand pane shows the data for the module. The default display shows two columns, the "ID" column and the "text" column, the title of which is the module description. The ID is a unique identifier assigned by DOORS when an object is created. DOORS uses this identifier to keep track of the object and any other information that is associated with it, *e.g.* attributes and links. The text column displays the data like a document, showing a combination of the heading number, the heading itself and the text associated with each requirement.

DOORS provides a number of display options for formal modules as shown in Figure 9.5. In the *Standard View*, all levels of objects are displayed in a "document" format. Users can restrict the display level, *e.g. Outline* displays only headings, hiding all other object details. This result is similar to a typical document "table of contents". As stated earlier, the *Explorer View* is useful for seeing the structure of the module and for navigating to a specific object in the module.

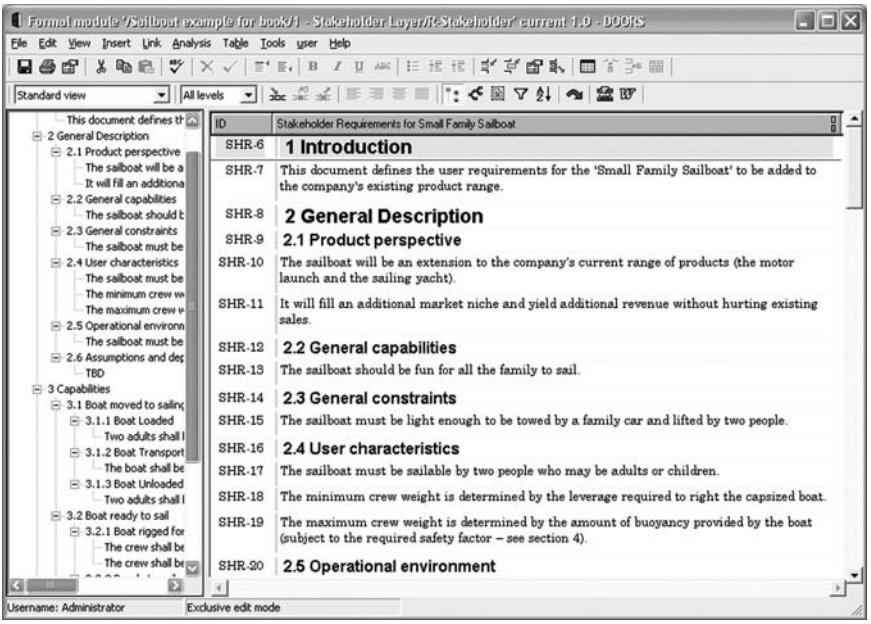

Figure 9.4 Formal module default display.

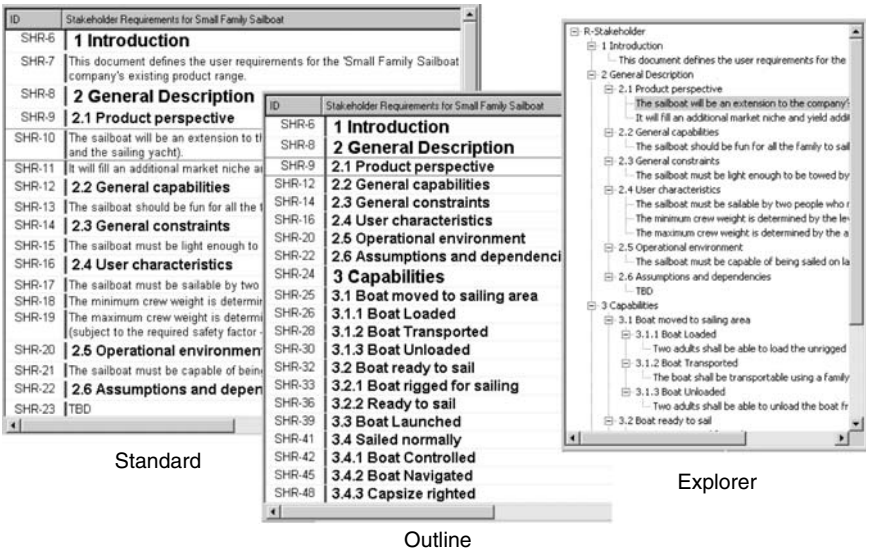

Figure 9.5 Formal module display modes.

Graphics mode, on the other hand, represents the display as a tree (Figure 9.6), which aids navigation through large data sets. The titles of the objects in graphics mode are based on the *Object Heading* and a shortened version of the *Object Text*.

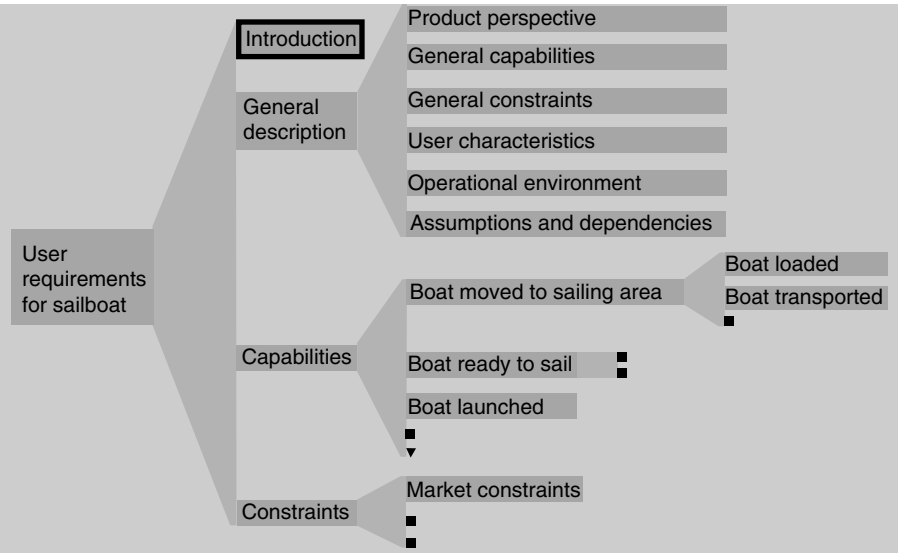

Figure 9.6 Graphics mode.

9.4.3 Objects

As we have seen in the previous section, within formal modules data is stored in objects. An object may be a block of text, a graphic image or even a spreadsheet created using another package. The standard view of a formal module display includes two columns and a number of visual indicators as described below.

As shown in Figure 9.7, the first column displays the *Object Identifier* assigned by DOORS. The Object Identifier is made up of two parts:

- a prefix (typically an abbreviation for the requirement set);
- the absolute number, supplied by DOORS.

The absolute number is an integer assigned sequentially (1, 2, 3, etc.) that serves as a key for each object, unique within the module.

The second column is known as the *Main* or *Text* column. It includes a composite three attributes, depending on contents:

- section number (*e.g.* 1, 2.1, 3.2.3), indicating the object's position in the module structure;
- object heading, providing a title for the object;
- object text, giving the full description of the object.

Object numbers are only displayed for objects that have been assigned an object heading.

Black lines above and below the object indicate the Current Object. Many functions relating to objects in DOORS modules, *e.g.* inserting a new object, pasting an object and moving an object are performed relative to the current object.

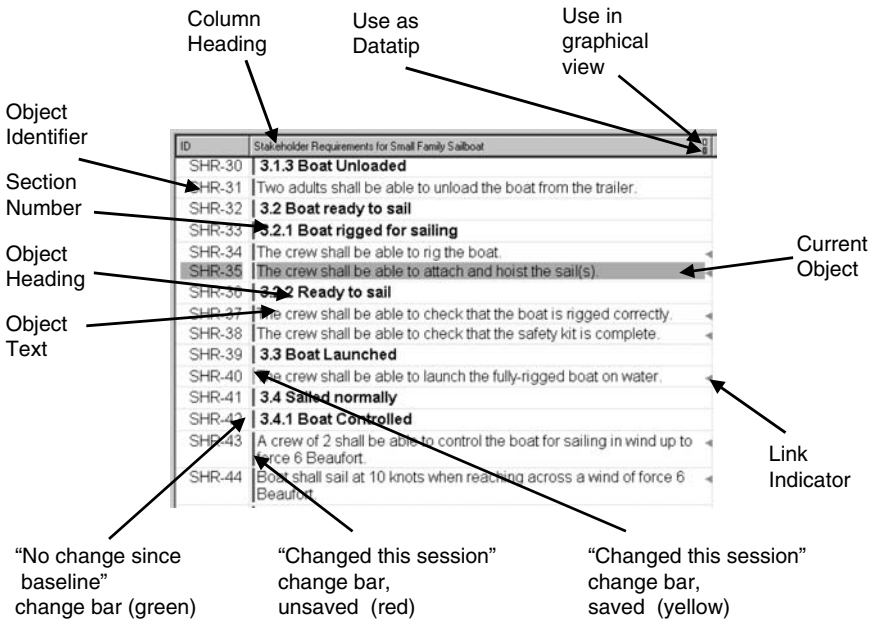

Figure 9.7 Displayed information.

Green, yellow and red *change bars* appear at the left edge of the text column. Green denotes an object that has not been changed since the last module baseline. Yellow shows that changes have been saved since the baseline and red indicates unsaved changes made in the current session.

Maroon and orange *link indicators* are displayed on the right-hand side of objects, which have relationships to other objects. The orange triangle pointing to the left indicates an incoming link and a maroon triangle pointing to the right indicates an outgoing link.

A DOORS formal module tree structure provides a simple, yet powerful, method of writing requirements. Requirements are often organized into a hierarchy, and so the graphics mode is a useful view.

Creating new objects in DOORS is simple – new objects are placed in one of two positions relative to the current object. Either:

- a new object is created as the next sibling of the current object with **Insert ▶ Object**, or
- an object is created as the first child below the current object with **Insert ▶ Object Below**.

This is shown in Figure 9.8.

In DOORS, facilities are provided for editing objects. For example, a DOORS tree can be modified by using the cut and paste functions. The *cut* operation removes the current object and all its children from the module display. This causes the rearrangement of the object hierarchy, collapsing the tree into the

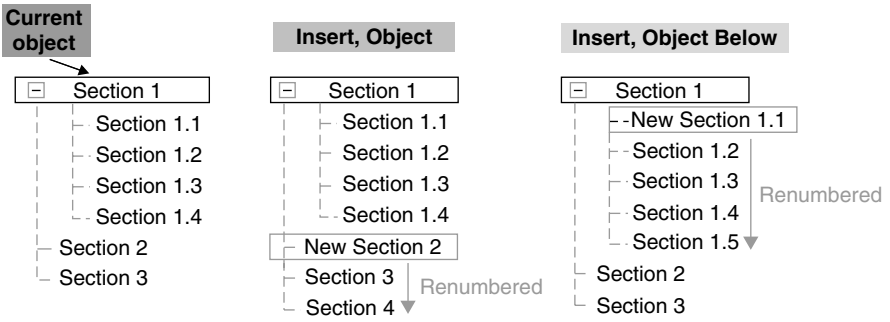

Figure 9.8 Creating objects.

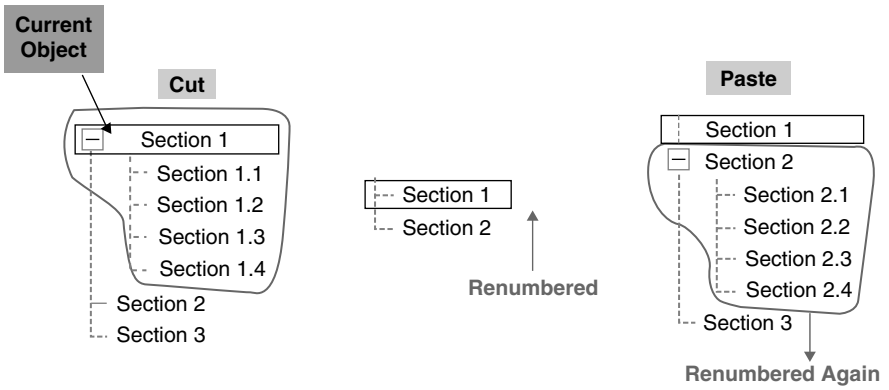

Figure 9.9 Cut and paste objects.

hole vacated by the objects that have been cut. This produces a renumbering of the remaining successor objects. The insertion point can then be identified for the objects that have been cut. Because of the nature of an object hierarchy there are only two possibilities. Objects are placed *after* as the next sibling to the current object or one level down as the first child of the current object. The former is shown in Figure 9.9.

9.4.4 Graphical Objects

Graphical objects in the form of embedded OLEs can be inserted into any text attribute in DOORS, in much the same way as OLEs are handled in Word, for instance. This allows pictures, diagrams, charts, documents, spreadsheets and many other kinds of information to be inserted into the requirements document in support of requirements statements.

9.4.5 Tables

In many cases, requirements or information associated with requirements is presented in tabular form. Tables can be created after or below an existing object, or

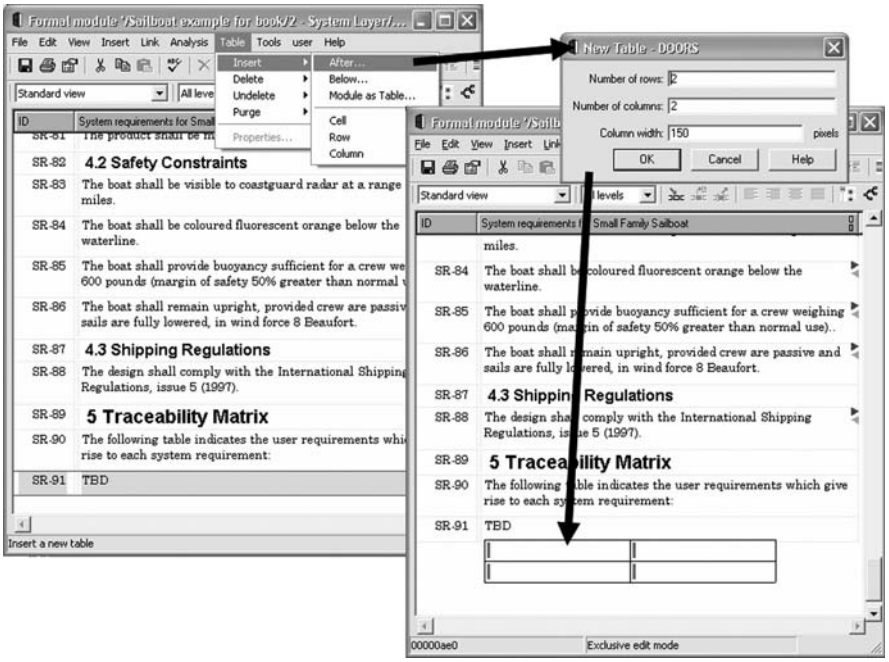

Figure 9.10 Creating a table.

at level one in an empty module. This is achieved by specifying the number of rows and columns required. The new table can then be inserted into the formal module as shown in Figure 9.10. Tables can be deleted as long as there are no links to any of the cell objects or to the table object.

9.5 History and Version Control

9.5.1 History

DOORS maintains a historical log of all module and object level actions that modify the contents of a module, its objects and attributes.

Every change that is recorded includes who made the change, when the change was made and what were the before/after states of the object and its attributes. The module history can be used to track every event in the life of a given module. The object history can be accessed via the change bar in the formal module window or it can be launched from the main menu. An example history window is shown in Figure 9.11.

9.5.2 Baselining

A baseline is a frozen copy of a module. They are typically created at significant stages of a project, *e.g.* a set of requirements is normally baselined immediately

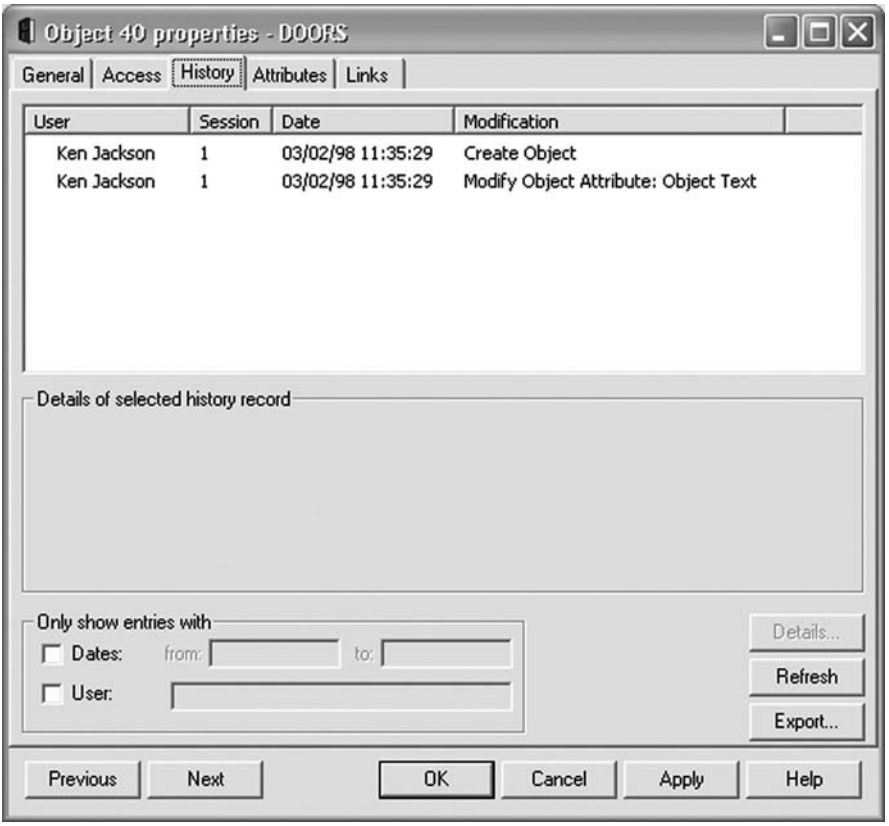

Figure 9.11 History window.

prior to a review, and then immediately after the resulting changes from the review have been incorporated. This allows the various states of the requirements document to be easily reproduced at any time. Baselines can be numbered and labelled in DOORS.

Baselines are read-only copies of a formal module and cannot be edited. When a module is baselined, all history since the previous baseline is stored with the newly created baseline and the history for the current version is cleared. The life history of a module is therefore stored across a series of baselines.

9.6 Attributes and Views

9.6.1 Attributes

Attributes provide a means to annotate modules and objects with related information. Module attributes are used to capture information about the module itself, such as its owner, document control number and review states. Object attributes are used to capture information about individual objects. Attributes

may be either system or user defined. System attributes automatically maintain critical information about a module or object, such as when it was created and by whom, whereas user-defined attributes may be used to capture any information required to support the user's requirements management process.

DOORS provides a variety of standard attribute types, known as *base types*, from which attributes may be defined, *e.g.* integer, real, date, string, text, user name. It is also possible to have user-defined attribute types.

Attribute information can be readily viewed and edited by creating columns. In this way, both on-screen and printable reports can be readily generated. Whereas an object may contain many attributes, a user is typically interested in viewing a subset of these attributes at one time. Columns may be created to show just the desired subset so that the user is not overwhelmed with information. Simply dragging and dropping the column header can reposition columns.

9.6.2 Views

DOORS provides a facility called *views* for looking at the same information in many different ways. Views are stored with modules and it is possible to create many views from a project's data. When creating views, the object and attributes which are to be displayed are specified. For example, you might wish to create a view that lets you see only those objects in the module whose "Priority" attribute has a value "High". A view then appears as a table, where each row contains one object and the object attributes that have been selected.

9.7 Traceability

Traceability is managed in DOORS through the use of links between objects.

9.7.1 Links

A DOORS link is a connection between two objects. One property of a link is directionality; all links have a direction, from source to target. To represent data relationships a link is created, thus enabling the user to visualize information as a network rather than just a tree. Although links have directionality, DOORS provides the capability to navigate in either direction through the path that a link creates between two objects. Hence it is possible to trace the impact of changes in one document on another document, or trace backwards to indicate the original thinking behind a decision.

DOORS provides a variety of methods for creating and maintaining links. Individual links can be created using drag-and-drop between two objects (usually in different modules). Whole sets of links can be created in other ways. For instance, the c*opy and link* function can copy a whole set of objects, and link each copy to its original.

Links are indicated along the right hand side of the main column in the standard view of a formal module by triangular link icons. An icon that points to the left represents incoming links and the opposite for outgoing links.

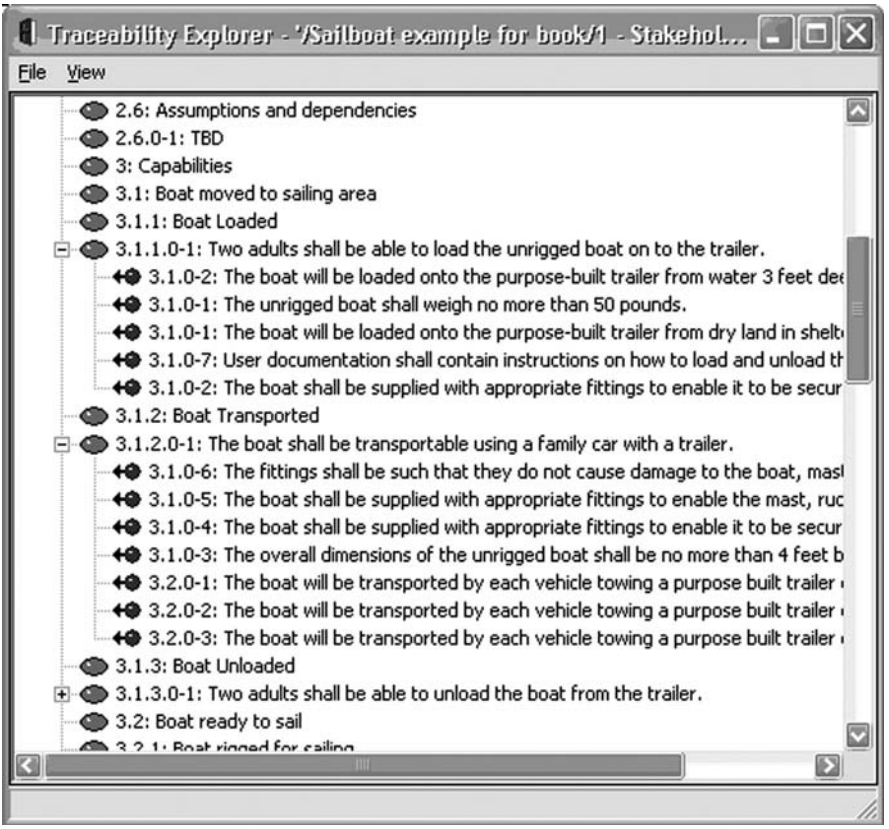

Figure 9.12 Traceability explorer.

9.7.2 Traceability Reports

There are a number of ways in DOORS of creating traceability reports, on screen and on paper. The simplest on-screen traceability tool is through using **Analysis ▶ Traceability Explorer**, which uses a Windows-style interface to allow the user to explore traceability across multiple documents in a single window. This is illustrated in Figure 9.12.

Another way of constructing an on-screen report (which can subsequently be printed) is by adding traceability columns to a view. These columns can display data about linked objects from other documents. They are created using **Analysis ▶ Wizard**, which guides the user to select which links and which attributes of linked objects are to be displayed. Traceability columns are completely dynamic and are updated as new links are created or as the distant data is changed. Through this means, data from several documents can be brought together into a single report, on-screen or printed on paper.

Figure 9.13 shows an example of a view that contains a traceability column. The view lives in the current module, which is the stakeholder requirements, and the column shows data from the system requirements module by following the

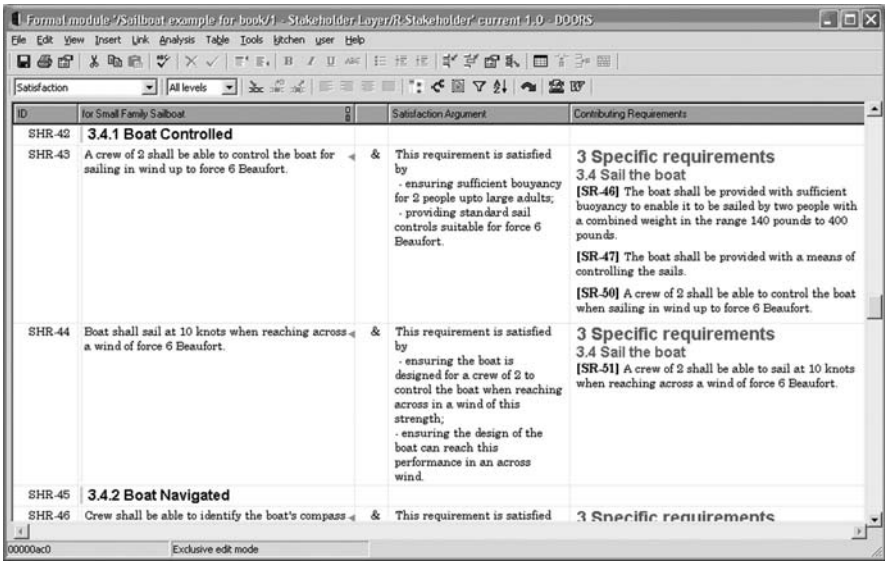

Figure 9.13 Traceability columns showing requirements satisfaction.

incoming links. Rich traceability is used in the example, and the columns are as follows:

- the Stakeholder requirement identifier (from current module);
- the main column showing the stakeholder requirement heading/text (from the current module);
- the rich traceability combinator (an attribute of the stakeholder requirement in the current module);
- the satisfaction argument (an attribute of the stakeholder requirement in the current module);
- a traceability column entitled "Contributing Requirements" showing several attributes of system requirements linked to the stakeholder requirement. The object identifier of the system requirement is shown bold in square brackets, followed by the text. In addition, the section headings of each system requirement are shown to give essential context within the system requirements document.

Figure 9.14 shows a traceability column from the other end of the same links, *i.e.* from the system requirements document back to the stakeholder requirements. In this case, the outgoing links are traversed and information is displayed in the column entitle "Originating Requirements". There is no column for the satisfaction arguments.

It is common for requirements documentation to include traceability matrices showing the relationships between documents. Through the use of traceability columns in views, DOORS avoids the need to create and maintain such matrices manually.

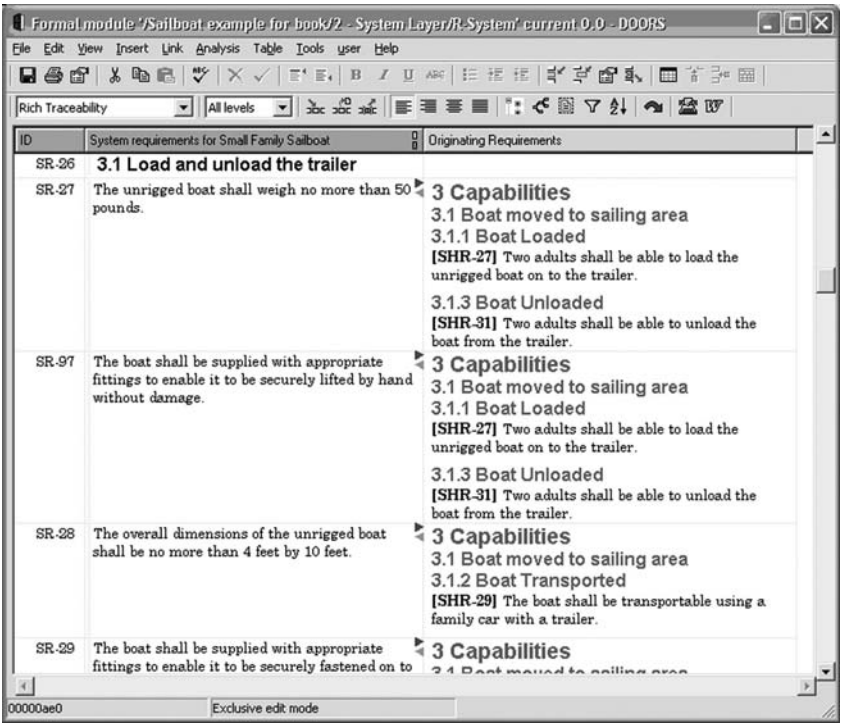

Figure 9.14 Traceability column on outgoing links.

9.8 Import and Export

The capability of exchanging information between DOORS and the tools is highly desirable. This can range from importing legacy information into DOORS and for exporting DOORS information to external tools for publishing and other purposes.

In project development, the ability efficiently and reliably to import and organize large quantities of information is often necessary. However, the variety of storage formats and platforms and the inconsistencies in data structures can make this a real challenge. DOORS provides a wide range of import tools to support this activity and in particular in relation to documents, tables and databases. For example, Figure 9.15 shows how to input from Word into DOORS. This is achieved by opening a Word document and exporting it to DOORS, using the **Export to DOORS** button – a module name and description needs to be supplied before the file is exported from Word and imported into DOORS.

The document is imported into DOORS with the same structure as the Word Outline view, so Heading 1 text becomes an object at level 1 in DOORS. Paragraph breaks are used for delimiting the content of each object.

Similarly, DOORS supports many export formats to provide a convenient method of transferring DOORS data to other desktop tools. As an example, consider exporting from DOORS to Word as shown in Figure 9.16.

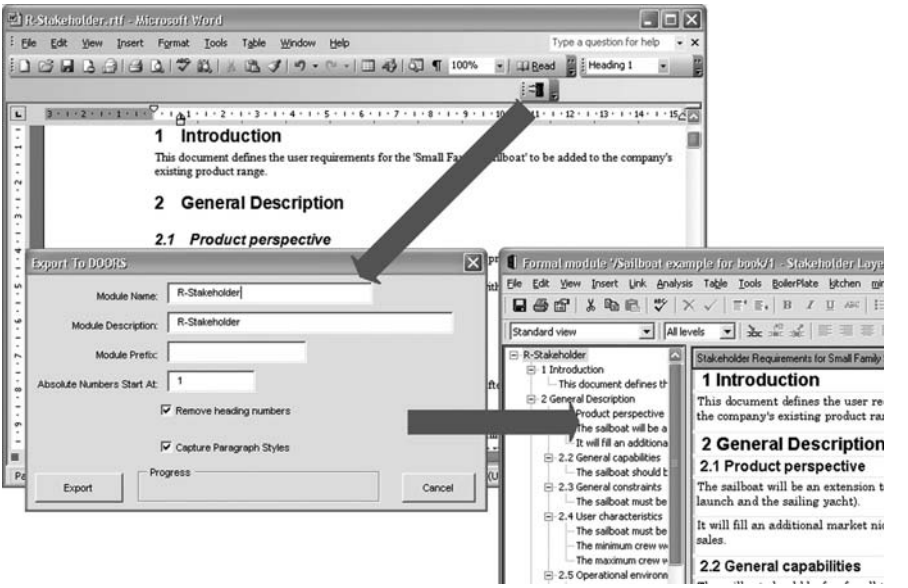

Figure 9.15 Export from Word to DOORS.

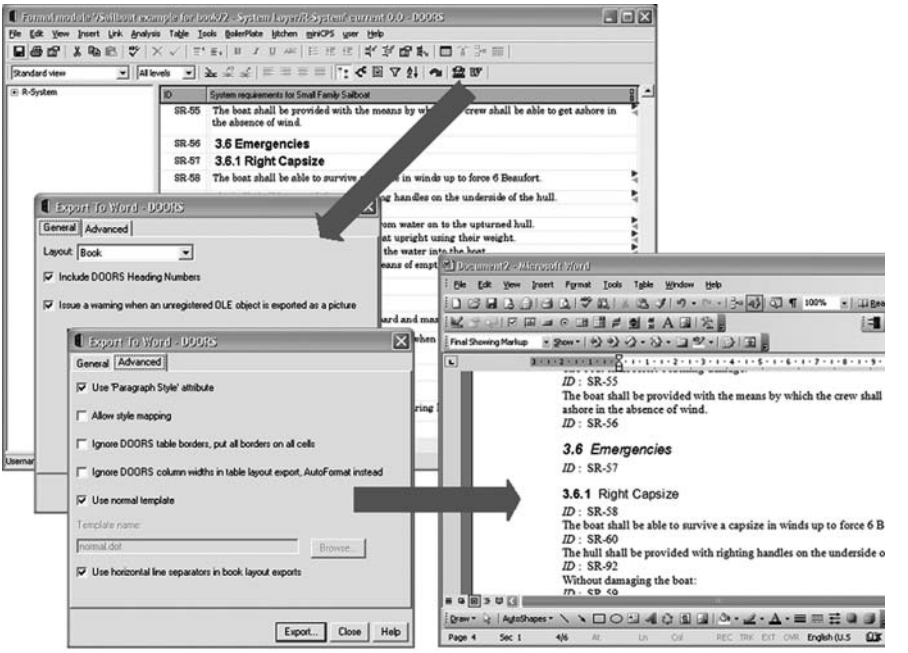

Figure 9.16 Export from DOORS to Word.

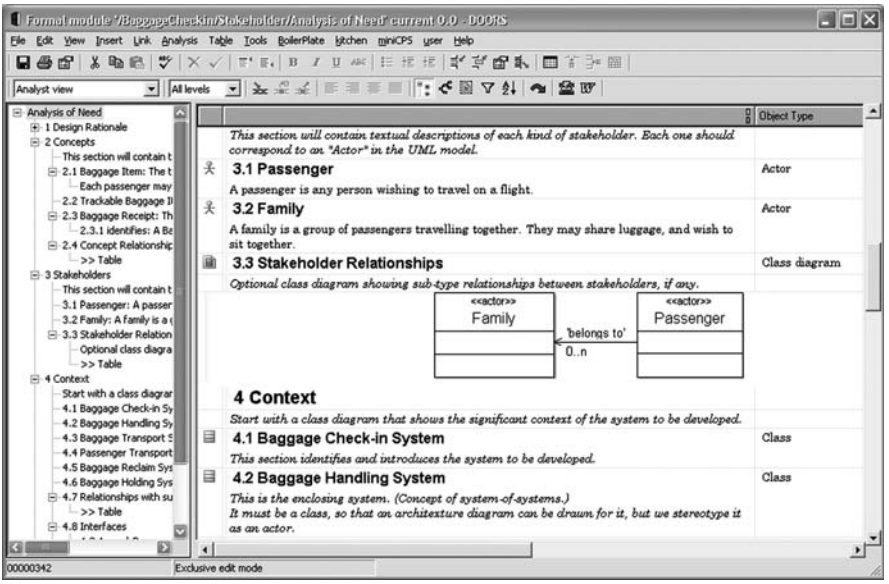

Figure 9.17 UML modelling in DOORS/Analyst.

This is the reverse of the previous operation. The Word document will have the same structure as the formal module, *i.e.* Object Heading 1 becomes Level 1, Headings become Word style Heading 1, and so on. Text is displayed in Normal style.

DOORS provides these types of import and export capabilities for a range of tools and formats, including RTF, Word, WordPerfect, Excel, Lotus, Access, Plain Text, HTML, PowerPoint, MS Project, Outlook and many others.

9.9 UML Modelling with DOORS/Analyst

DOORS/Analyst is an integration of DOORS with the Telelogic UML modelling tool TAU. It permits UML models to be created and diagrams to be presented within a DOORS module.

As requirements are analyzed, objects in a DOORS module can be labelled as UML elements, such as actors, classes and states. When a diagram is inserted into the DOORS module by activating the UML modelling tool, the DOORS objects so labelled are synchronized with elements that appear in the diagrams. The effect of this is to allow traceability of requirements into elements that appear in diagrams in UML.

Figure 9.17 shows a screen-shot of a DOORS module in which DOORS/Analyst has been used to label objects and insert a class diagram. Labelled objects are indicated by icons in the narrow column to the left of the main column and the type of UML entity is also shown in the "Object Type" column to the right.

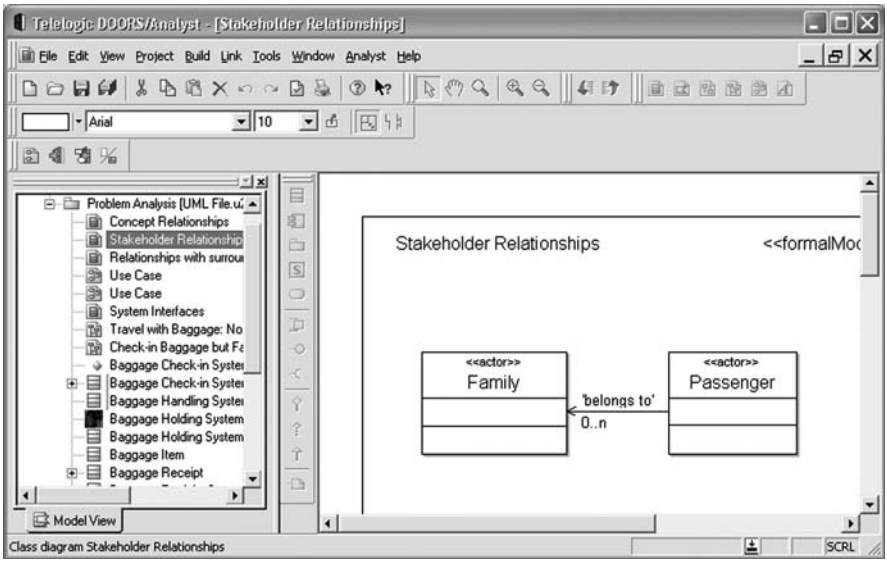

Figure 9.18 UML diagram editor in DOORS/Analyst.

Double-clicking on a diagram starts up the DOORS/Analyst diagram editor, shown in Figure 9.18. Saving changes to the model causes information to be resychronized into the DOORS module.

9.10 Summary

A brief overview of a requirements management tool, DOORS, has been given. The example used shows the application of some of the principles used in the book, *e.g.* instantiations of the generic process in layers and rich traceability.

DOORS/Analyst is also introduced as an example of how modelling tools can compliment requirements management tools.

The same principles can be applied and implemented in other requirements management tools. Even if one is just using a word processor, the disciplines described within the covers of this book will be found beneficial.