From Document To Database: Modernizing Requirements Management

J. Giajnorio[1] and S. Hamilton[1]
[1] General Electric (GE), Peterborough, Ontario, Canada

Abstract

The creation, communication, and management of design requirements are central to the successful completion of any large engineering project, both technically and commercially. Design requirements in the Canadian nuclear industry are typically numbered lists in multiple documents created using word processing software. As an alternative, GE Nuclear Products implemented a central requirements management database for a major project at Bruce Power. The database configured the off-the-shelf software product, Telelogic Doors®, to GE's requirements structure. This paper describes the advantages realized by this scheme. Examples include traceability from customer requirements through to test procedures, concurrent engineering, and automated change history.

1. Background

On an engineering project, requirements are the main form of documented design communication between customers and suppliers, between detailed designers of each subsystem, and between designers and testers. As a tool for designers they are intended to capture the desired functionality of the system, set out constraints, and define interfaces. Customers use system level requirements as a contractual document stating their minimum expectations of what constitutes an acceptable end product; suppliers use the same requirements to bound their scope of work. Requirements are the basis for test specifications, which are typically written with a one-to-one correspondence between requirements and test cases.

In GE's experience, poor requirements can result in schedule delays, increased costs, and quality misses due to rework and scope creep. Studies have found that it is at least 14 times more costly to fix a problem discovered in the test phase than if the problem had been found in the requirements definition phase [1].

In the Canadian nuclear industry, standard practice is to write requirements as numbered "shall" statement lists in a word processing program to create a text document. GE Nuclear Products' methods and experience are representative of the industry. On a large project this could involve tens of thousands of pages of requirements, from the top-level system requirements down to the detailed requirements for each component. It is very difficult to manage multiple people working on a document simultaneously in a word processing program, which necessitates that requirements be split into individual documents that can be authored by only a few people. Frequently references must be made within one requirement to requirements in another document, such as the parent requirement. These references must be entered and maintained manually as a document acronym paired with the requirement number. Typically the reference is uni-directional, meaning that the child requirement references the parent, but the parent does not reference its children. Compounding this problem is that the automatic numbering feature of the word processing software is relied on for requirement references. This means that a requirement number can change when additions or deletions are made, making it very difficult to maintain external references to specific requirements. Manually numbering requirements is cumbersome and runs the risk of assigning the same number twice.

Software development is generally held to more detailed standards for requirements writing than other types of subsystems. The CANDU® Computer Systems Engineering Centre of Excellence software engineering standards prepared by AECL and OPG specify, for example, that real-time process software evaluated as Category II must demonstrate traceability. This is defined as identifying each requirement uniquely and explicitly, and demonstrating the mapping of requirements to the design description by such means as a coverage matrix. Producing these matrices manually in word processing software is a time-consuming task that by its nature becomes exponentially more onerous and error-prone as the size of the project increases. The same standards also specify that a detailed revision history be maintained within each requirements document. [2]

Other industries with strict requirements standards, such as aerospace and medical, have grappled with the limitations of the document methodology for requirements and moved to a database structure. A database supports multiple users working simultaneously. It is particularly well suited to storing relationships between objects, a feature that can be used to achieve traceability in requirements management. NASA has experimented with the use of database tools for requirements on many of its major projects and found that a custom requirements tool based on a relational database was superior in terms of metric capabilities to a word processor, spreadsheet, or a generic database solution [3].

2. Database implementation

2.1 Selection of software

To improve quality, reduce design time, and achieve the traceability increasingly required by nuclear industry standards, GE evaluated several off-the-shelf requirements management software tools in 2004. Thirty different tools were compared against the following criteria listed in descending order of importance:

- Provides requirements traceability
- Is easy to learn for Microsoft® Word users
- Allows collaboration
- Capable of exporting requirements to a text document
- Applies to all engineering disciplines (mechanical, instrumentation & control, and software)
- Provides verification procedures
- Provides configuration management of requirements (baselining)
- Reputable vendor
- Tracks requirement status

- License Cost
- Implementation Cost

A Quality Function Deployment (QFD) chart was used to assess which product best satisfied the needs of the engineering group as defined by the criteria. Each criterion was assigned an importance value and each product was given a score of low, medium, or high for each criterion. By multiplying the importance value of each criterion by the product's score and summing, a total score for each product could be computed. Figure 1 shows the overall results of the QFD. As the highest scoring product, Telelogic Doors® (hereafter referred to as Doors) was selected as GE's requirements management tool.

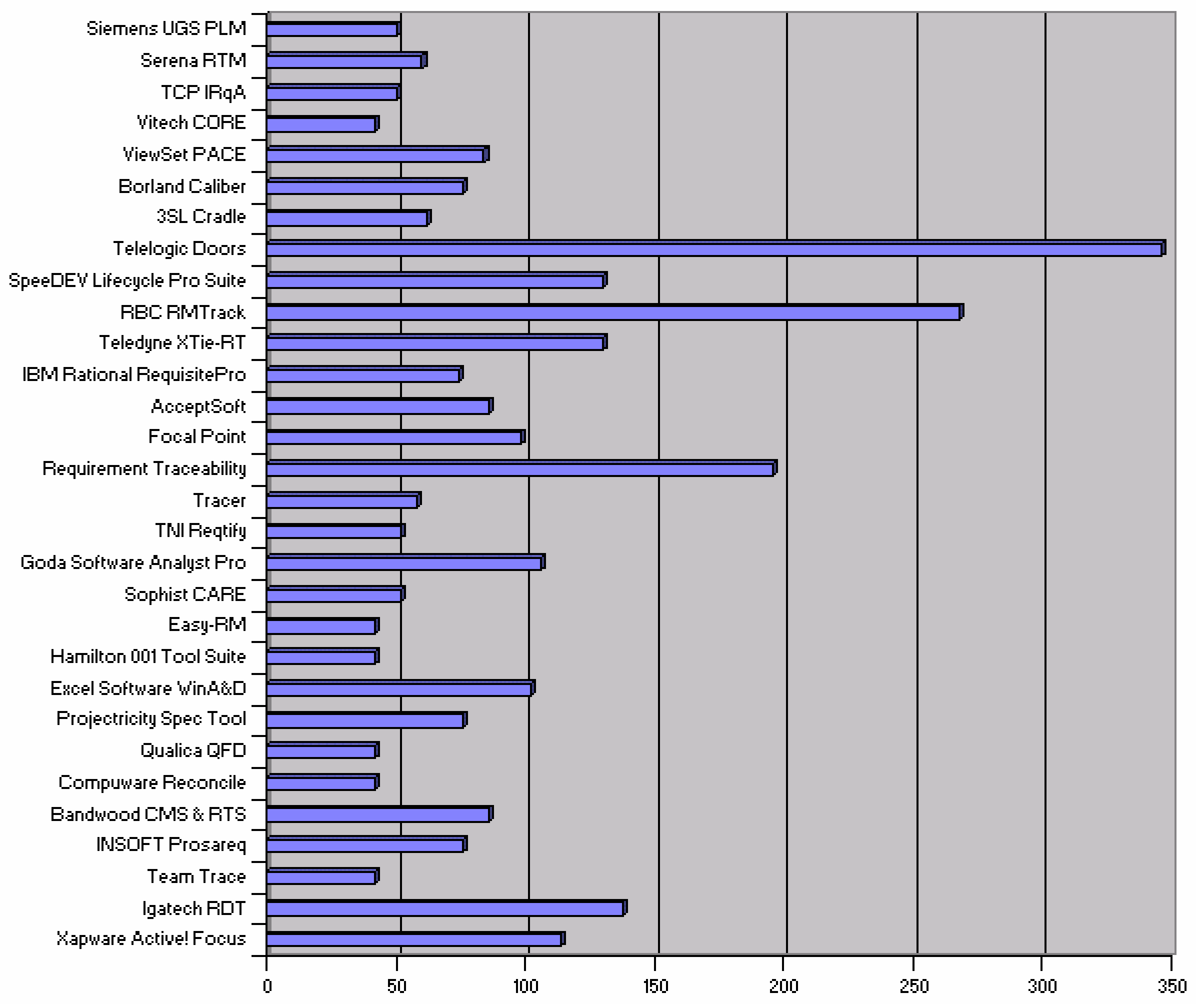


Figure 1 Comparison of requirements management tools.

2.2 Design process mapping

A cross-functional team was drawn from across the engineering department to receive training in the Doors software product and to plan an implementation strategy. Part of the training session involved mapping out a generic requirements structure as a template to apply to future projects. The main requirement groups on a typical project were identified, as were the links between them. The necessity of imposing a standard structure with strict link definitions onto a project from the outset was a point repeatedly emphasized by the course instructor. The ease with which links between requirements can be added in a database paradigm can result in designers adding excessive, redundant, and conflicting linkages. Such a scenario reduces the effectiveness of filter and search tools, hampers traceability, and ultimately negates the advantages of a database over a collection of text documents.

NASA's Software Assurance Technology Centre (SATC) published an example of a large project at their organization that was compromised by poor control of their requirements management tool. A retrospective examination of the requirements by a custom metrics analysis program developed by SATC uncovered deficiencies in the number of detailed requirements, the completion of requirements (TBDs), and test coverage. The root cause was the decision to structure the requirements in the database according to organizational divisions. This discouraged information sharing, resulting in redundant requirements, circular links, and inconsistent terminology. The study concluded that advanced planning of how requirements are structured in the tool, followed by enforcement of the structure, is key to the successful use of this type of product [3].

The requirements structure established by the GE implementation team is shown in Figure 2. The basic arrangement of requirements, architecture specification, and interface definitions at the system level is replicated at the sub-system level and would be repeated to the lowest design level as appropriate for the scale of the project. Five possible link types are identified; the structure further specifies between which modules these links are allowed. Clear traceability is achieved from customer requirements to component test specifications. A Doors project with the illustrated module configuration and linkset definition was created as a generic template.

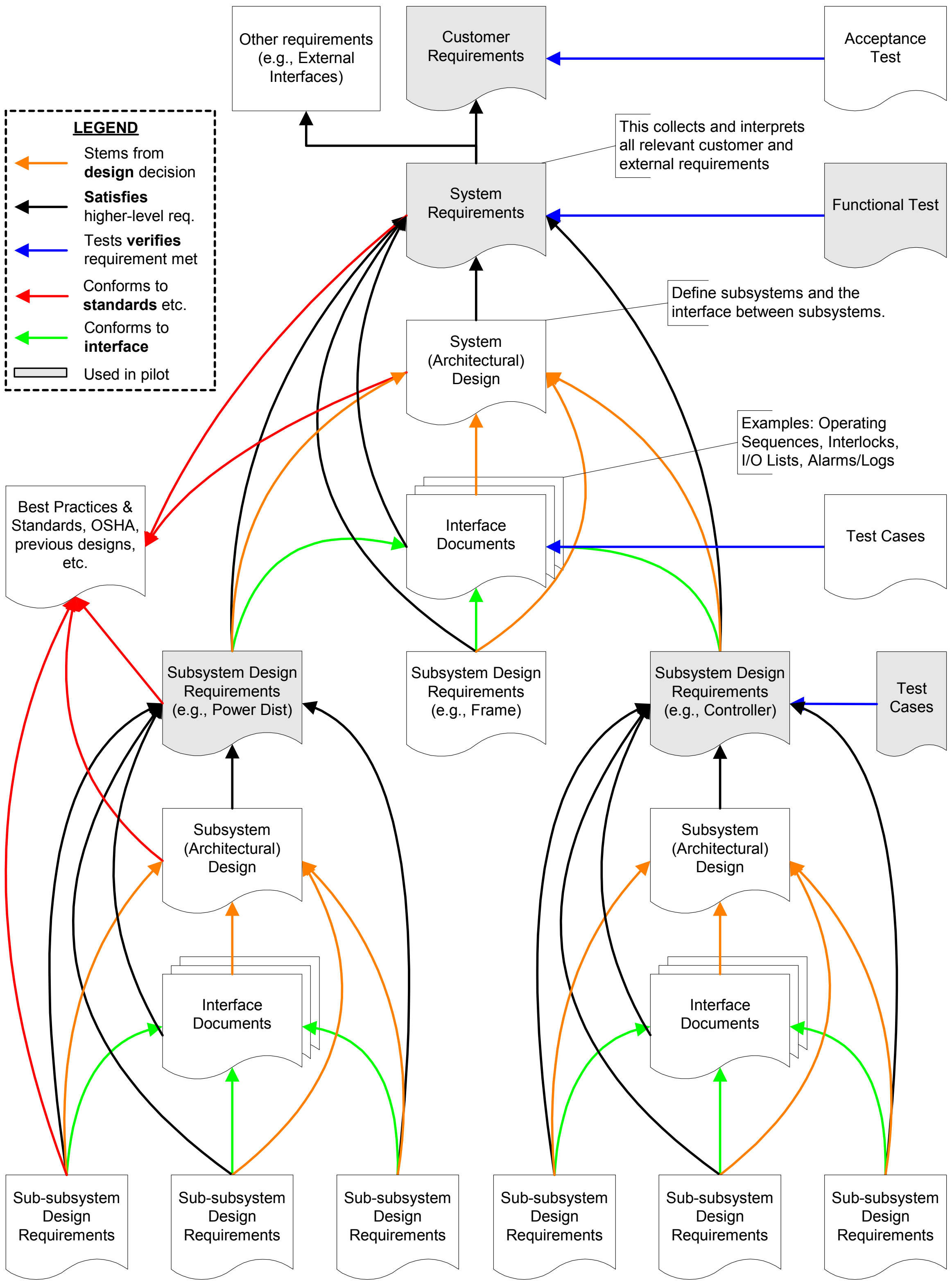

Other requirements (e.g., External Interfaces)
Customer Requirements
Acceptance Test
LEGEND
Stems from design decision
Satisfies higher-level req.
Tests verifies requirement met
Conforms to standards etc.
Conforms to interface
Used in pilot
This collects and interprets all relevant customer and external requirements
System Requirements
Functional Test
Define subsystems and the interface between subsystems.
System (Architectural) Design
Examples: Operating Sequences, Interlocks, I/O Lists, Alarms/Logs
Best Practices & Standards, OSHA, previous designs, etc.
Interface Documents
Test Cases
Subsystem Design Requirements (e.g., Power Dist)
Subsystem Design Requirements (e.g., Frame)
Subsystem Design Requirements (e.g., Controller)
Test Cases
Subsystem (Architectural) Design
Subsystem (Architectural) Design
Interface Documents
Interface Documents
Sub-subsystem Design Requirements
Sub-subsystem Design Requirements
Sub-subsystem Design Requirements
Sub-subsystem Design Requirements
Sub-subsystem Design Requirements
Sub-subsystem Design Requirements

Figure 2 Generic requirements structure for database.

3. Pilot project

Following the Doors training and the establishment of the requirements structure, the implementation team selected a project on which to pilot the use of Doors as a requirements management tool. The criteria for selection were that the project be large enough to have a significant number of requirements, yet limited in the number of engineering disciplines involved. The team expected a number of changes to the software configuration and workflow processes, especially in the early stages. This would have been difficult to manage with a larger group.

The selected pilot project was a major system rehabilitation project at Bruce B. In addition to satisfying the above criteria, the software portion of the project was not considered safety-critical, so the development process was not tightly constrained by standards such as OASES. This freed the team to experiment with the project structure and workflow.

The system under upgrade was originally put into service in the early 1980s, and the design requirements had not been maintained over its lifetime. At the start of the project, Bruce Power provided GE with a new set of customer requirements.

3.1 Project structure

For the pilot project, the authors were the project administrators, responsible for setting up the structure of Doors and managing user privileges. This was an ideal arrangement as the authors were also the primary system designers on the project. Since none of the other project members had been trained in Doors, the project structure was fully created in Doors before work started on any of the requirements documents. This minimized rework resulting from modifications to the project structure once the project was underway.

The project structure was a limited version of that shown in Figure 2. The shaded modules are those that were implemented for the pilot project. Many of the sections in the figure were unnecessary, or inapplicable, due to the limited scope of the project. For example, interface documents between the System Design and Subsystem Design modules were not warranted, as the interfaces were simple enough to be fully captured within the System Requirements.

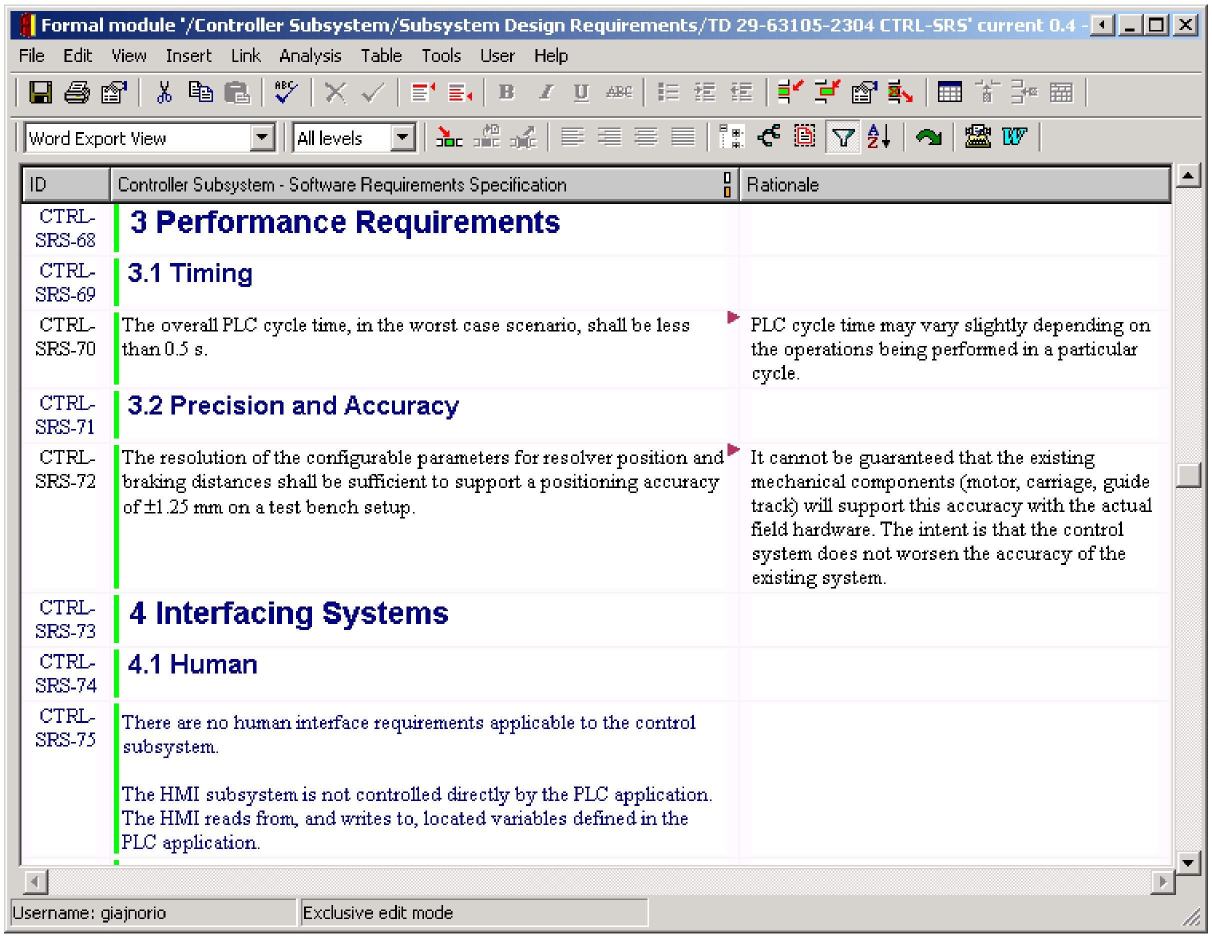
Each box in Figure 2 is represented in Doors by a "formal module", which is a collection of related requirements. In the final project deliverables, each formal module was exported as a requirements text document. Links between each module are captured in "links modules". These store the different link types specified in Figure 2, but are never edited directly. The links between modules are made while editing the requirements.

In addition to the requirement text, a number of attributes common to each module were created:

- **Requirement Type:** Each requirement was labeled as Active, Exception, Cancelled, or Description.
- **Rationale:** An explanation of why the requirement is present.

- **General Comments:** Used by designers to enter comments relevant to each requirement.
- **Review Comments:** Used by document reviewers to enter comments for each requirement, and by GE staff during formal design reviews to note customer comments.

A Doors feature known as "views" was used to facilitate different ways of working with the requirements modules. A view in Doors saves the complete display configuration, including the attributes displayed, column size and arrangement, and any filters in effect. Each view was configured for a particular set of tasks; for instance, a view created for working with the document would show all requirements, while a view created for reviewers omitted cancelled and excluded requirements. A limited number of default views, common to each module, were created. Users also had the ability to define custom views. Figure 3 shows an example of a view set up to facilitate exporting the document to Word format. This view shows only the attributes that would appear in the exported document.


Formal module '/Controller Subsystem/Subsystem Design Requirements/TD 29-63105-2304 CTRL-SRS' current 0.4 -
File
Edit
View
Insert
Link
Analysis
Table
Tools
User
Help
Word Export View
All levels

ID	Controller Subsystem - Software Requirements Specification	Rationale
CTRL-SRS-68	3 Performance Requirements	
CTRL-SRS-69	3.1 Timing	
CTRL-SRS-70	The overall PLC cycle time, in the worst case scenario, shall be less than 0.5 s.	PLC cycle time may vary slightly depending on the operations being performed in a particular cycle.
CTRL-SRS-71	3.2 Precision and Accuracy	
CTRL-SRS-72	The resolution of the configurable parameters for resolver position and braking distances shall be sufficient to support a positioning accuracy of ±1.25 mm on a test bench setup.	It cannot be guaranteed that the existing mechanical components (motor, carriage, guide track) will support this accuracy with the actual field hardware. The intent is that the control system does not worsen the accuracy of the existing system.
CTRL-SRS-73	4 Interfacing Systems	
CTRL-SRS-74	4.1 Human	
CTRL-SRS-75	There are no human interface requirements applicable to the control subsystem. The HMI subsystem is not controlled directly by the PLC application. The HMI reads from, and writes to, located variables defined in the PLC application.	

Username: giajnorio | Exclusive edit mode

Figure 3 Sample Doors view.

3.2 Design stage

Before starting the design stage, the authors entered the customer requirements into Doors. This allowed the authors to experiment with the views for creating and working with requirements before rolling them out to all designers on the project. As additional users were added, they were trained by the authors in one-on-one sessions. The average time required to train a user to the point where he or she was able to add to and modify requirements without assistance was 45 minutes.

The designers linked each requirement to its parent requirements as it was created. Users created custom views that listed only "orphan" requirements, essentially creating an automatically updated list of requirements that needed to be linked. With each requirement correctly linked to its parent, users were able to immediately determine when a higher-level requirement had been modified, and which requirements were affected by the change.

Two of the attributes described in section 3.1 facilitated review of the requirements. The “General Comments” attribute provided a place for designers or anyone else viewing the document to add suggestions, questions, or placeholders for future changes. The “Review Comments” attribute provided a place to store feedback from internal or external design reviews. Once either type of comment was addressed, it was deleted. Doors automatically maintains a full history of all changes, so users can easily see the time and date when a comment was resolved.

During design reviews the requirements were reviewed on-line with the Doors interface projected on a screen, viewed by all attendees. All issues were recorded directly into the “Review Comments” field for each requirement. The history for each requirement therefore provided a complete record of when each comment was entered and resolved, and by whom. This ensured that all comments from design reviews were completely and formally traceable.

When a requirements module was issued as a draft or final document, it was exported to Word using a script developed in the proprietary Doors extension language. Immediately before or after export the current module state was captured using a Doors feature known as "baselining". Baselining saves a read-only copy of the module, and assigns it a version number. This is a common feature of most requirements management systems, and it allows users to track exactly what was issued without the need to retain paper copies of documents. Doors provides built-in features to automatically compare baselines with respect to each other or against the current state of the module.

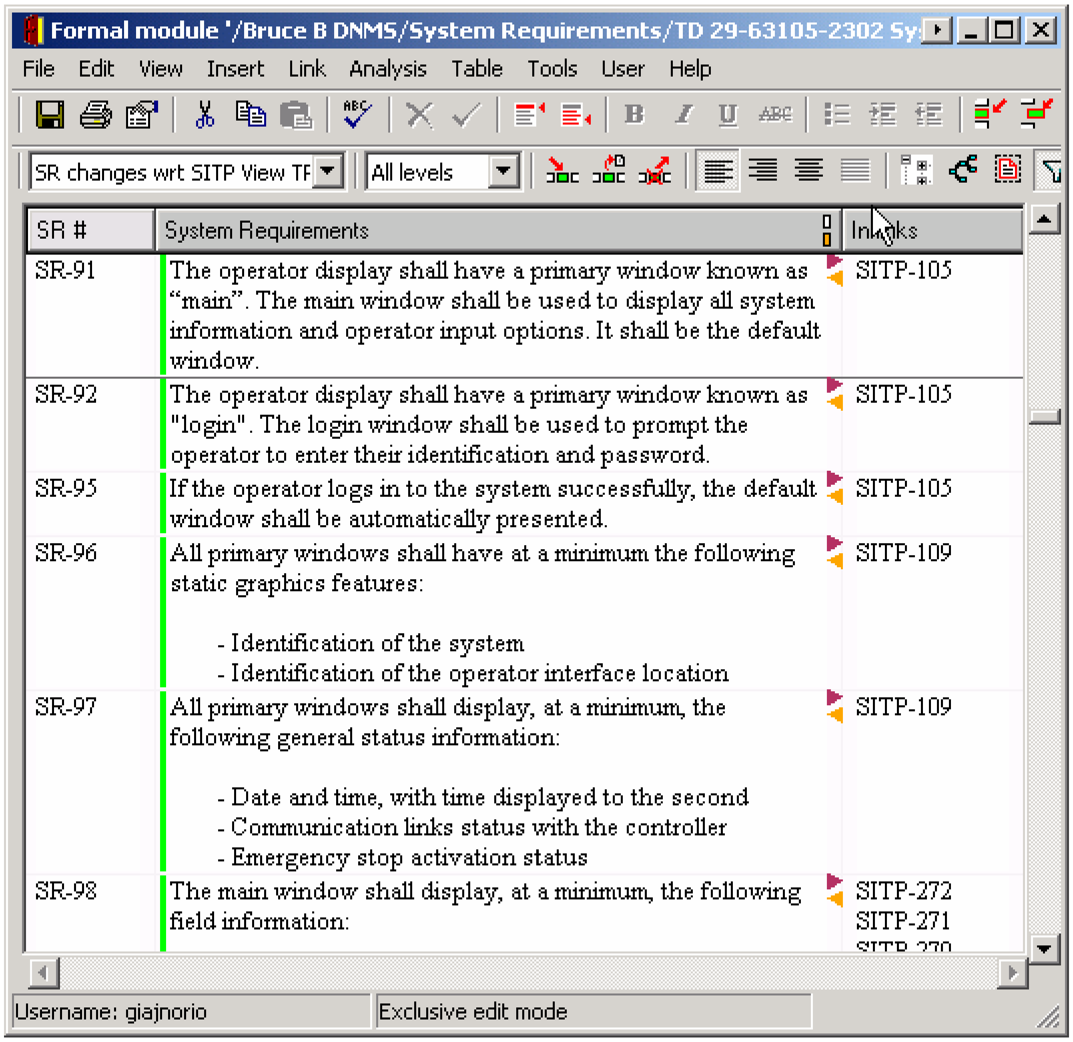
3.3 Test stage

Doors was used for the software integration testing phase of the project, which tested all software against the system-level requirements. A custom set of attributes was created for the test plan:

- **Test Objective:** Purpose of the test.
- **Test Procedure:** Detailed instructions.
- **Pass/Fail:** Result of the test.

- **Deficiency:** Detailed description of a test failure.

Each test plan was linked to the requirements it was intended to verify. This allowed a user viewing the system requirements to automatically generate a report showing how each requirement was verified. Figure 4 shows an excerpt from a more complex report, which showed only requirements modified after a certain date, and identified the associated test numbers (SITP-XXX). This report was used by the tester to ensure that all recently modified or added requirements were covered. A view created for printing the test plan automatically generated the list of requirements each test covered.


Formal module '/Bruce B DNMS/System Requirements/TD 29-63105-2302 Sy
File Edit View Insert Link Analysis Table Tools User Help
SR changes wrt SITP View TF
All levels
Username: giajnorio
Exclusive edit mode

SR #	System Requirements	In-links
SR-91	The operator display shall have a primary window known as "main". The main window shall be used to display all system information and operator input options. It shall be the default window.	SITP-105
SR-92	The operator display shall have a primary window known as "login". The login window shall be used to prompt the operator to enter their identification and password.	SITP-105
SR-95	If the operator logs in to the system successfully, the default window shall be automatically presented.	SITP-105
SR-96	All primary windows shall have at a minimum the following static graphics features: - Identification of the system - Identification of the operator interface location	SITP-109
SR-97	All primary windows shall display, at a minimum, the following general status information: - Date and time, with time displayed to the second - Communication links status with the controller - Emergency stop activation status	SITP-109
SR-98	The main window shall display, at a minimum, the following field information:	SITP-272 SITP-271 SITP-270

Figure 4 Sample view showing linked tests for each system requirement.

Test results were not entered into the system in real time, as the testers did not have access to a Doors workstation in the test lab. Rather, the test plans were printed and filled out by hand during the testing process, and results were entered into Doors upon completion of testing. For future projects, it is hoped that a Doors workstation could be made available in the lab, as this would automatically provide a complete history of all tests, including the tester's identity and the time and date when each test was performed.

3.4 Advantages realized

One of the most significant advantages of the database-oriented system was that every user had, at any given moment, full access to the most recent versions of all requirements. A document-based system relies on users to place their most recent versions into a central repository, which will only be current if no documents are checked out for editing by other designers.

Another major advantage was the ability to automatically trace the impact of any requirement change. Since each requirement was linked to its parents, and integration test cases were also linked, users could easily generate a report showing which requirements would be affected by a modification. In a document-based system, traceability is performed through manually updated tables, if at all. If there are several levels of documents, with each requirement potentially linked to several others at upstream and downstream levels, generating a traceability report becomes a significant manual effort.

Similarly, reports could automatically be generated in Doors showing the full traceability from test cases to customer requirements, proving that all customer requirements had been addressed and verified. Evidence of the value of this feature to a customer is that once they were shown a traceability report showing how their requirements had been addressed, the customer on this project immediately requested that this report be included in the output documents.

The ability to quickly navigate links between requirements proved advantageous during design reviews, as it allowed reviewers to quickly view the parent or child requirements. With paper-based requirements, navigating to a parent or child requirement involves looking up the current requirement in a cross-reference table, then searching for the listed parent or child in one or more separate documents. Using the Doors system, this navigation took place on-screen with a few mouse clicks. The ability to enter comments directly into the database was another significant improvement to the review process.

Although the requirements were all held in the database, GE and the customer's engineering procedures still required that paper copies of requirements documents be signed for issue. Although an extra step was required to export the requirements in each Doors module to Word format, exporting ultimately proved to save time. Doors contains built-in functionality for exporting documents to Word; the authors extended this feature using the Doors scripting language to export to Word templates with formatting specific to the project. Although some minor manual formatting was required following export, it was very minimal. The export procedure took an average of only 15 minutes from start to finish. In a document-based system, a great deal more time can be spent formatting the document; with the database-oriented system, users could focus exclusively on the requirement content. Similarly, in a document-based system, users must manually ensure that requirement numbers are never duplicated and do not change from version to version. This work is critical but time-consuming, with a very high potential for error.

The Doors scripting language used to customize the Word export is extremely powerful, and can control other Windows applications such as Word or Excel. The possibilities for using this language in the future to extend the system's functionality, or to customize the output, are significant.

3.5 Challenges encountered

Although the requirements management system described here provided considerable advantages over a document-based process, the project did encounter certain challenges. Many of these were specific to the Doors product, and could potentially be resolved by upgrading the software, adding custom scripting, or moving to a different product or custom system.

Many users commented that the Doors user interface, although easy to learn, was difficult to work with. The interface is set up primarily to handle short, single-paragraph requirements and does not handle large passages of text well. Parts of these passages are often skipped while scrolling through the document, making it difficult for users to locate or edit a particular piece of text. Also, while editing large passages the cursor frequently jumps below the visible area of the screen. More generally, users must navigate a number of menus and dialog boxes to perform most operations, including common tasks such as creating links between modules.

Because external contractors were doing some of the work on the pilot project, the implementation team planned to use the Doors import/export functionality to allow the contractors to work on off-line copies of the requirements. This strategy ultimately proved to be infeasible. Users encountered no difficulties in working with the exported requirements, but the software failed to merge most changes when importing modified requirements. Instead, modified requirements were identified as being new, with the resulting numbering conflicts preventing a successful import.

This highlights a general disadvantage of database-oriented requirements management tools. Unless the software provides excellent import and export mechanisms, it can be difficult for someone without access to work within the system. Manual effort is required to provide exports from the database and to import changes. Word documents, in comparison, use a near-universal format that can be read and edited by anyone.

Most requirements management tools, including Doors, automatically assign requirement numbers in order of creation. Once a requirement is created, the number is never re-used, even if that requirement is immediately deleted. Although this frees users from spending time maintaining requirement numbers and eliminates duplication, the resulting requirement numbers are often not sequential. During the pilot project, Bruce Power commented that this made it difficult to locate a particular requirement number. In this case, adding a requirements index to each exported document, automatically generated by a custom Doors script, was a satisfactory solution.

4. Conclusion

Migrating to a requirements management database tool has great potential to improve the efficiency and quality of the design process. The pilot project using Doors for requirements and test specifications was successful and realized several gains in traceability, concurrent engineering, and change control. Adopting a database solution presents challenges, however, that are best overcome by careful attention to the selection and implementation of the particular tool. The experience of using Doors in the pilot project uncovered several minor limitations of the software that were not insurmountable, but reduced efficiency. Other products may exhibit more severe drawbacks.

Configuring a Doors project, in particular the linksets, is not trivial and there is a risk that, if it is done incorrectly, not only will the advantages of a requirements management tool not be realized, but the results could become even more cumbersome than a paper solution. It is important to emphasize that these types of tools neither evaluate the quality and accuracy of the requirement text, nor assess the relevance of the links between them. They do not replace designers, but rather assist them.

GE is investigating the feasibility of rolling out Doors to all new projects in engineering. The first step is to put in place data collection to measure the effectiveness of this process change. There is a lack of good data on our current document-based requirements process, which would make it difficult to quantify performance improvements. There is presently a project in progress at GE to map our entire design process, establish time benchmarks for design activities, and record anecdotal evidence of designers' challenges and frustrations.

5. References

[1] Boehm B. "Tutorial: Software Risk Management", *IEEE Computer Society Press*, 1989

[2] CANDU® Computer Systems Engineering Centre of Excellence, "Standard for Software Engineering of Category II Software", Standard CE-1002-STD, Revision 0, 1990

[3] Hammer T.F., Huffman L.L, Rosenberg L.H., "Doing Requirements Right the First Time!", Software Technology Conference, Utah, April, 1998