

Requirements Management of the

"Dashboard Software Metric

Visualization for SQR" Project Using

Telelogic's Doors Software

April 2003

Prepared By: John James Willson
Prepared For: Prof. Armin Eberlein
Westmost 010 Course

Table of Contents

ABSTRACT	3
1. INTRODUCTION	4
1.1 REQUIREMENTS MANAGEMENT	4
1.2 WHY SMV - SOFTWARE METRIC VISUALIZER	4
1.3 APPROACH TO APPLYING DOORS TO SMV	5
2. DOORS.....	5
2.1 EVALUATION CRITERIA OF REQUIREMENTS MANAGEMENT SOLUTIONS	5
2.2 ADDITIONAL NON-FUNCTIONAL REQUIREMENTS MANAGEMENT	5
2.2.1 <i>Installation of Doors</i>	5
2.2.2 <i>Features expected and not provided</i>	6
2.3 SUBSEQUENT APPROACH TO SMV	6
3. SMV PRIOR TO DOORS REQUIREMENTS MANAGEMENT	7
3.1 DOCUMENTS AND REQUIREMENTS OF SMV	7
3.2 TRACEABILITY AND TRACKING	9
3.3 CONFIGURATION MANAGEMENT	10
4. SMV WITH DOORS.....	11
4.1 CONVERSION TO DOORS	11
4.2 IDENTIFYING AND CAPTURING REQUIREMENTS.....	14
4.3 FEATURES OF THE DOORS TOOL USED	14
4.4 REQUIREMENTS PROJECT FLOW	15
4.5 TRACING REQUIREMENTS AND LINKING.....	15
4.6 VERIFICATION OF REQUIREMENT COMPLETION.....	16
4.7 NOT ENOUGH LOW LEVEL REQUIREMENTS	17
4.8 HISTORY OF REQUIREMENT CHANGES AVAILABLE	17
4.9 DXL - NOT FOR EVERYONE	18
4.10 INTERFACING OUT AND DOCUMENTATION	18
4.1 <i>Microsoft Office Suite Products</i>	18
4.2 <i>Use the Web</i>	19
4.3 <i>Compiled Help</i>	19
5. CONCLUSIONS.....	19
5.2 INTERFACE IMPROVEMENTS	19
5.3 EXCELLENT COLLABORATION	19
5.4 CHANGE THE DOORS METAPHOR	21
5.5 OVERHEAD VERSUS EFFICIENCY.....	21
5.6 PUTTING THIS DOCUMENT IN DOORS.....	21
APPENDICES.....	22
A. REFERENCES (THIS DOCUMENT)	22
B. REFERENCES (SMV EXPORTED FROM DOORS).....	22

List of Figures

FIGURE 1 SMV DIRECTORY STRUCTURE	7
FIGURE 2 REVIEW OF VISUAL BASIC CODE	9
FIGURE 3 RELATIONSHIP OF REQUIREMENTS TO SOFTWARE PRODUCTS WITHOUT DOORS.....	10
FIGURE 4 DOORS DIRECTORY STRUCTURE.....	11
FIGURE 5 EXAMPLE START OF 11 PAGE DOORS USER REQUIREMENTS DOCUMENT	12

FIGURE 6 INITIAL DOORS SMV DOCUMENTS.....	13
FIGURE 7 DOORS CAPTURING OF SMV REQUIREMENTS	14
FIGURE 8 USE OF DOORS WIZARDS.....	14
FIGURE 9 USE OF DOORS IMPORT/EXPORT OF RICH TEXT.....	15
FIGURE 10 DOORS TRACING REQUIREMENTS THROUGH TO CODE	16
FIGURE 11 USE OF DOORS ANALYSIS.....	17
FIGURE 12 DOORS BUGS AND LOCKS.....	18

List of tables

TABLE 1 DOORS DESCRIPTORS APPLIED TO MICROSOFT OFFICE PRODUCTIVITY TOOLS	6
TABLE 2 SMV REQUIREMENTS WITH-OUT DOORS	8
TABLE 3 SUMMARY IMPROVEMENTS WITH DOORS.....	19

ABSTRACT

This report describes the application of Telelogic's Doors 6.0 SR1 Enterprise Requirements Suite product to a software development project entitled "Dashboard Software Metric Visualization for SQR" (SMV). The software project consisted of all aspects of requirements engineering but was completed without any automated requirements engineering management software. The software project artifacts directly concerned with requirements engineering were subsequently managed using the Telelogic's Doors product. This management could then be used as a baseline to compare the usefulness of the Doors product and to make comments concerning the product.

After the introduction, the following pages describe the Doors product and how it was going to be evaluated with SMV. Subsequently, SMV requirements engineering and management pre-Doors is then described. SMV using the Doors product is described in some detail. Finally conclusions about the usefulness of Doors to a software development project are presented.

Although there are problems of installation and operation of Telelogic's Doors product, the benefits of using the product and automated requirements management far outweigh the problems. The problems are enumerated through out the report and from that perspective this report may appear negative. The report also highlights the clear advantages for even a small project like SMV that accrue with automated requirements management. In particular, the insights down to the code level are possible through the linking and maintenance of requirements through out the systems development life cycle. These insights provide for not just contractual observations; they also provide assistance in doing projects themselves.

As a result of retrofitting SMV requirements and 3 cycles of a spiral life cycle including code through Doors, many comments about the SMV project itself could be made. Requirements were identified and never met. Code was written that had no link to any requirement. Impacts of changing requirements spawned by the stakeholders were never estimated or prioritized. Just done. The quality of software projects could be enhanced through even informal requirements management and greatly improved through automation of the requirements engineering process. It is not that there was not any requirements management in SMV. It is that by automating and highlighting requirements management additional insights into the project itself are possible.

1. INTRODUCTION

The following paragraphs provide some comments concerning requirements management. Following those comments, a discussion of why the SMV project was used in this report is provided. Finally, the approach to using the Telelogic's Door product with SMV is discussed.

1.1 Requirements Management

One comment on requirements management that is not often made, is that requirements management provides a way of achieving closure on a given project. By agreement not to pursue requirements or by completion of all prioritized requirements, closure on a project can be proven. However, requirements engineering is not just concerned with project closure. It is also concerned with:

- The capturing of project requirements.
- Tracing requirements through project artifacts.
- Measuring and identifying the impacts or changing requirements with in a project. And,
- The answering of whether all requirements of a given project have been met.

One advantage of an automated requirements management product is that it can provide a baseline for measuring project change. In the SMV project, the first prototype was considered a baseline. Indexing or numbering of requirements did not occur until the first prototype was built.

1.2 Why SMV - Software Metric Visualizer

In the SMV project, requirements management was not well done from the point of view of managing requirements. Requirements were successfully elicited and described both in writing and diagrammatically. However, requirement identification through numbering and tracing was not accomplished until well into the project.

The stakeholders of the SMV project consisted of sponsors (both academic and industrial), a project manager, developers, and a user community or group. These stakeholders subsequently became some of the views used in the Doors / SMV review.

SMV was a small project consisting of 100,000 lines of code (100 KLOCs) and 50 or so requirements. This made it readily possible to use Doors through all aspects of requirements engineering management as a microcosm of a larger project. Doors was applied to SMV artifacts from requirements elicitation through to coding.

SMV was a master's project. It consisted of building a working source line visualizer with 3 prototypes completed, end-user evaluation and numerous artifacts. Artifacts included a shareware working SMV run time disk; PowerPoint slides; Adobe documents; Work documents; Visual Basic code; Install Shield code and DLLs; and market place, end user, and developer requirements

SMV was chosen because it was a contained project in which to apply Doors.

1.3 Approach to applying Doors to SMV

The approach to applying Doors to SMV consisted of converting selected requirements documentation and artifacts to Doors. Evaluation of Doors while doing the conversion. Then trying multiple Doors features to address how SMV could be helped with respect to requirements management both historically and for future development. Through out the above activities, documentation of what was occurring was completed for this report.

2. DOORS

There were many expectations of using Doors. A summary of Doors expectations is that Doors:

- Stores requirements using industry standards and templates.
- Adds attributes and or columns to requirements for sorting and filtering
- Links requirements with project artifacts and maintains the traceability chain both with fan-in and fan-out to SMV requirements
- Creates views of requirements by stakeholder need-to-know, and
- Handles changing requirements in an automated way.

This section of the report provides evaluation criteria to requirements management solutions. In addition it provides how requirements both functional and non-functional are met in SMV possibly using Doors.

2.1 Evaluation Criteria of Requirements Management Solutions

A major personal evaluation consisted of ease of use and rapid learning of a tool such as Doors with out the use of hardcopy or compiled help files and/or training. Other evaluations from industry include:

- How Doors captures requirements;
- What infrastructure that Doors uses to support requirements management;
- How does Doors support requirements flow through out the SMV project;
- How are requirements tracked, traced and their impacts measured;
- How does Doors use the SMV artifacts that were not contained in the Doors environment;
- How does Doors communicate across the SMV user community; And
- What is Doors look-and-feel in say the Windows networking environment.

2.2 Additional Non-Functional Requirements Management

The evaluation criteria above is more related to the functional needs of a requirements management solution. However there are also some non-functional requirements that should be used for evaluation. How a product such as Doors is installed and some of its possibly missing features are also causes for evaluation.

2.2.1 Installation of Doors

There were problems associated with the installation of Doors mostly related to security and the Windows environment it was going into. (paths in particular including the license location and registry took time to set up) These problems were readily overcome by exploration and multiple installation attempts; however, Doors did not install the way many other software products install in the Windows environment and took some shoe horning.

2.2.2 Features expected and not provided

A more severe problem from the point of view of evaluation of Doors were the features expected and not provided (perhaps in this version of the software).

Direct import from Windows required a higher skill level than anticipated for SMV Windows based artifacts. A work around was developed for implementing SMV requirements artifacts but it was a non-obvious work around. In particular, Doors table like structure and/or metaphor seemed to be the problem with Words table objects.

The following table describes the Doors / Microsoft Office Productivity Suite as guessed by the author.

Table 1 Doors descriptors applied to Microsoft Office Productivity Tools

Doors	Microsoft Office Productivity Suite
Attribute	Field or Column
Object (OLE look-a-like in MS terms)	Record or Row
Module	Table or Worksheet
Project	File
Proprietary Data base / Server	SQL Server / Access

Many of the problems of using Doors with SMV could be ascribed to the above table and its related metaphor. Tables in Word did not import correctly to Tables in Doors. Exporting of Tables in Doors would crash the Doors environment if not exported using rich text format. Using the Word format, the Word styles of SMV did not readily translate to vanilla Doors. All of these problems were overcome, but the learning curve was steeper than expected. Doors appeared to be database centric and that the database is proprietary. However, all information is retained in the Doors database and most of the clashes between the Doors look-and-feel and the Windows look-and-feel appear related to the Doors database. It is likely that the database has limited SQL capabilities since insertion of SMV items into the database was akin to insertion into many enterprise resource packages.

2.3 Subsequent approach to SMV

Problems with simple creation with in Doors were few. However, SMV was being Doors retrofitted. So work arounds were developed. These work arounds actually proved the power of what Doors was capable of. For example, most industry standards were already formatted as templates in Doors. By using these

templates considerable number of Word style problems could be eliminated. This consisted of

- Exporting formatted shell documents.
- Cutting and pasting from SMV documents and artifacts into shells.
- Retaining Doors Word styles along with formatted shells.
- Deleting old shell documents in Doors.
- Importing revised Word documents into Doors. And,
- Making new documents in Doors the revised documents.

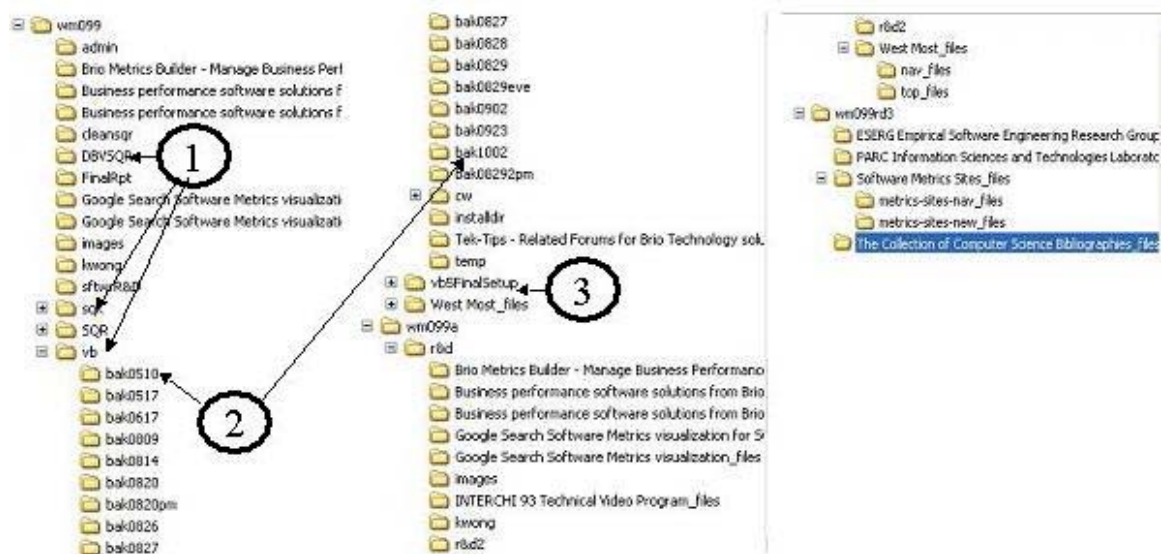
This work around worked for straight text; however, when importing Word tables into Doors, the Module and/or Object capabilities seemed to create undetermined results in the final Doors document.

3. SMV PRIOR TO DOORS REQUIREMENTS MANAGEMENT

This section describes requirements engineering prior to using Doors. After just reviewing the Doors product and not yet having applied the product, considerable improvements in SMV requirements engineering could be obtained.

3.1 Documents and Requirements of SMV

Figure 1 SMV Directory Structure



In figure 1, the directory structure of SMV is provided in a Windows environment. In point 1 indicated, note that there are several different kinds of SMV artifacts. In point 2 note the revision control system of SMV is manual and consists of milestone and/or key date backups of the entire software under development. Note in point 3, that there are builds of different SMV prototypes. Requirements are captured and *should be* linked in all areas. In table 2 following

the requirements are listed without using the Doors indexing system. These requirements are listed in the order of prioritization by the stakeholders.

Table 2 SMV Requirements With-out Doors

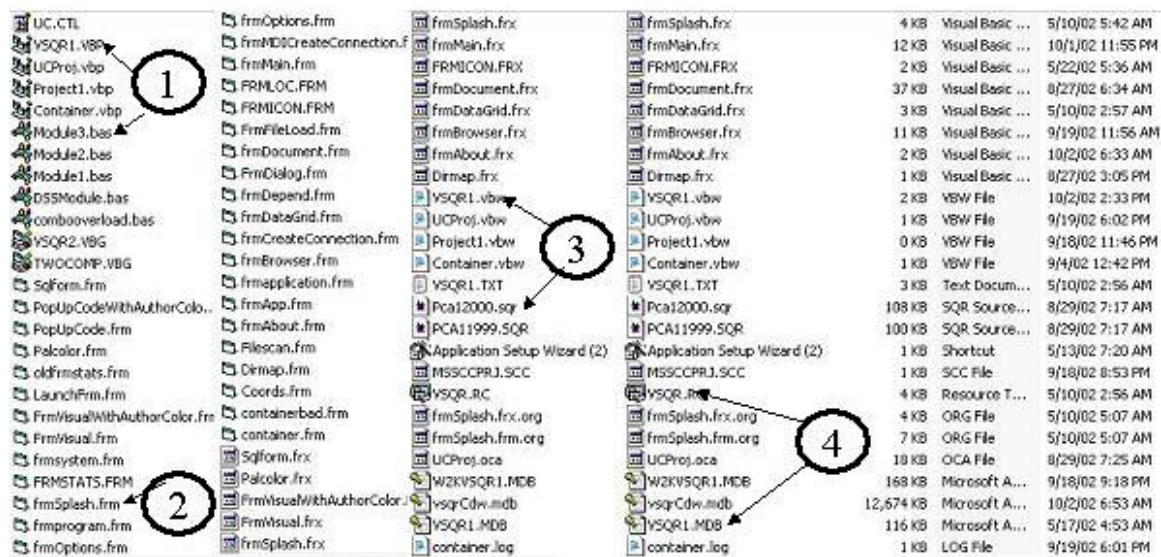
#	Requirement	Scale	Rationale
a1	Capture 'raw' statistics of code productivity	a	Core functionality of a visualizer. Presentation of these statistics is not necessary if they can be viewed appropriately.
a2	Drill down on visualized abnormalities	a	Provides a way of viewing the source code to inspect whether an abnormality is real or imagined
a3	Selectable targets of visualization including author, variables, language constructs, words	a	Provides a way of selectively visualizing.
a4	View change events consisting of change of author, code modification, other	a	Core operation of a visualizer. It major change events can not be viewed then you do not have a visualizer.
b1	Capture significant views for presentation and discussion	b	Provides a way of using visualization for human resource issues and/or maintenance issues.
b2	Easily retrieve SQRs for visualization	b	Provides a way of importing SQRs for visualization
b3	Highlight abnormal change events	b	Provides an automated way of viewing change events
b4	Interface to version control software such as PVCS, MKS, Rational, other	b	Provides an easy way of capturing change events of interest.
b5	Provide ad hoc reporting of statistics not just pre canned statistics	b	Provides a way of retrieving statistics of interest by passing out-of-the-box visualizations. Allows for more custom visualizations.
b6	Provide feedback on the visualizer operation	b	Assists a visualization user to make appropriate use of the visualizer.
b7	Provide rapid learning of visualizations including legend and titling	b	Assists in moving up the learning curve faster.
b8	Provide snap shot capability for before and after timed views	b	Provides a way of measuring changes in terms of percentages at a macroscopic level.
b9	Set colour codes for author, for language construct, for variable	b	Provides a way of distinguishing statistics of interest in a visual form.
b10	View a whole SQR application not just a program	b	Provides a way of managing application report suites
b11	View programmer language style	b	Provides a way of viewing particular developers way of using the language.
c1	Export visualizations to other office productivity tools such as Visio, Word, other	c	Provides a way of enhancing the visualization.
c2	Interface to developer tools such as TextPad, Brio Report Builder, other	c	Provides an easy way of making changes after viewing a change event.
c3	Provide thin client for visualization to the web	c	Provides a way of managing SQR development in multiple geographic locations.
c4	Set level of view such as an elevation ascend/descend	c	Provides for scalability of view. Results in settable levels of granularity.
c5	View consistency of programmer style	c	Provides a way of viewing language style independent of kind of language construct.

c6	View history of visualizations including archiving of significant visualizations	c	Provides a way of making routine a series of visualizations say from year to year.
d1	Convert visualizations into statistics or statistics into visualizations	d	Provides another media for understanding underlining change events.

3.2 Traceability and Tracking

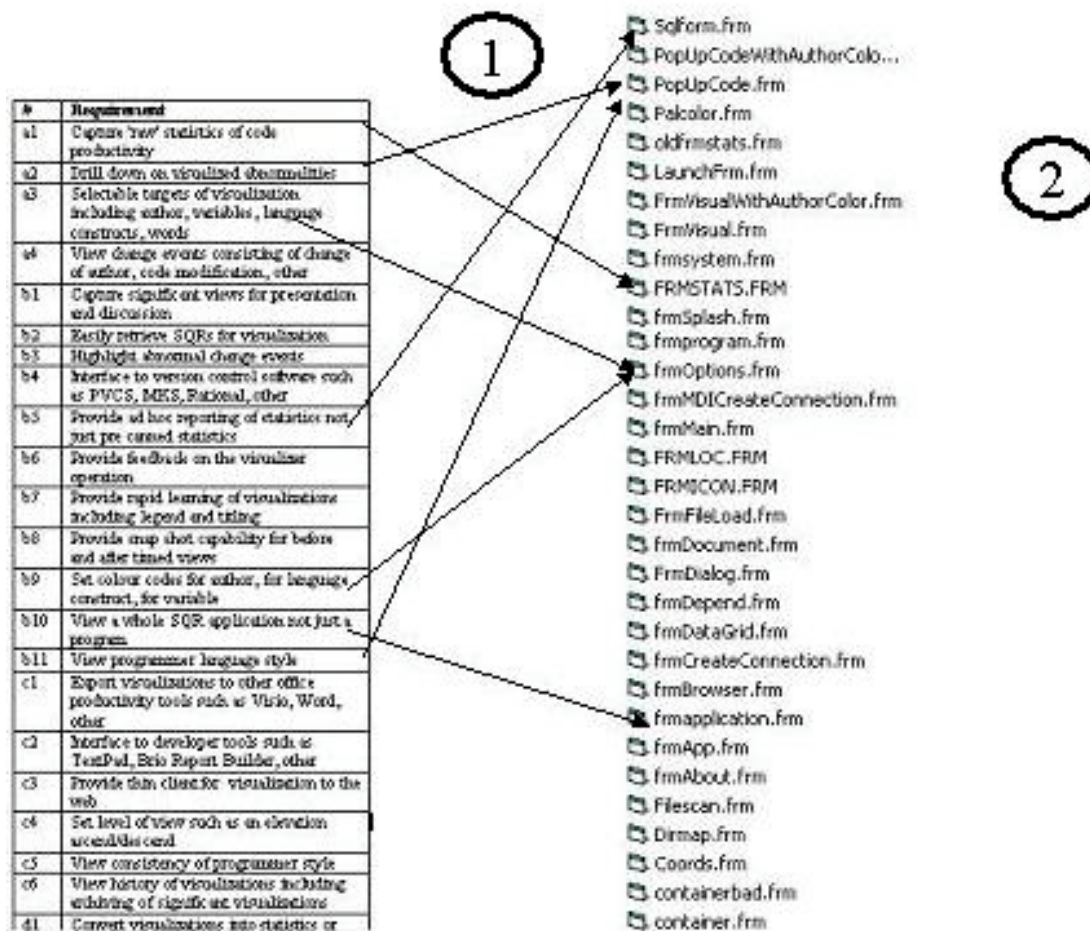
An important concern of software development is the relationship of the code to requirements. The evaluation copy of Doors that was used did not provide interface modules into revision control systems. In any case, SMV followed a homegrown revision control system as demonstrated in the directory structure.

Figure 2 Review of Visual Basic Code



In figure 2, a view of Visual Basic code artifacts is provided. In particular, note the file extensions frm and bas. These are forms with code and subroutine code. SQR is another languages source code artifact and an extension. Note that mdb is an Access database extension. So what is the relationship between say the requirements of SMV and the code of SMV? This relationship will be explored in the next figure. But through out this report, requirements will be linked from elicitation until coding. Hence the mentioning of code extensions.

Figure 3 Relationship of Requirements to Software Products Without Doors



The above figure relates requirements previously given (1) with code of SMV(2). Later the same relationship will be provided in Doors. However, this relationship was never done in the original SMV project at this level of detail. Only at the end user level of detail.

3.3 Configuration Management

Versioning builds, prototypes, and code control in SMV consisted of the directory structures previously shown. Requirements were never linked across directory structures or across artifacts. Requirements were linked only in the views of the project manager and the developers. This did not include the user community or the user group testers. SMV was presented at user conferences; however, requirements validation was never shown.

4. SMV WITH DOORS

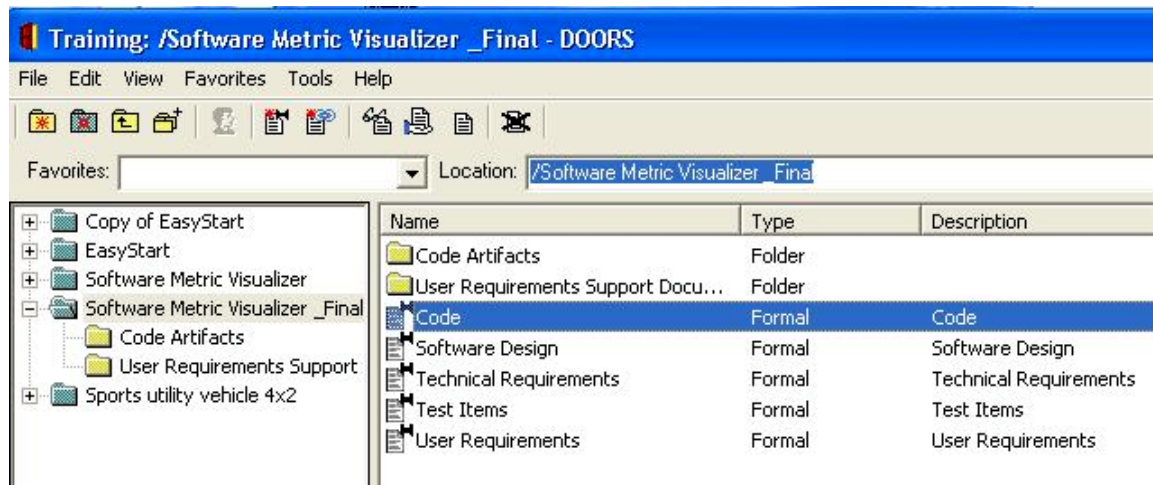
This section of the report is the main body of original requirements engineering work. It consists of a collection of activities performed and comments made when applying Doors to SMV.

Doors is document oriented within its proprietary database as opposed to say graphic oriented. (Not used with SMV was UML graphic techniques. Comments on whether Doors handles UML linking other than use cases can not be made in this paper) So many of the images used in this report would be special cases in Doors. In the demonstration package used for SMV there was not any import of Microsoft PowerPoint presentation file structures possible. It is expected that this exists but no work arounds were developed for the version of Doors provided.

Templates within Doors were excellent for standards building and meeting. These templates provided information content and structure. However, the standards did not always translate into Microsoft Word styles and required good editing in order to match for Doors import. Indeed Microsoft Word documents that were created independent of the given standards and had their own style sheets were difficult to import into Doors, especially Word documents with imbedded tables.

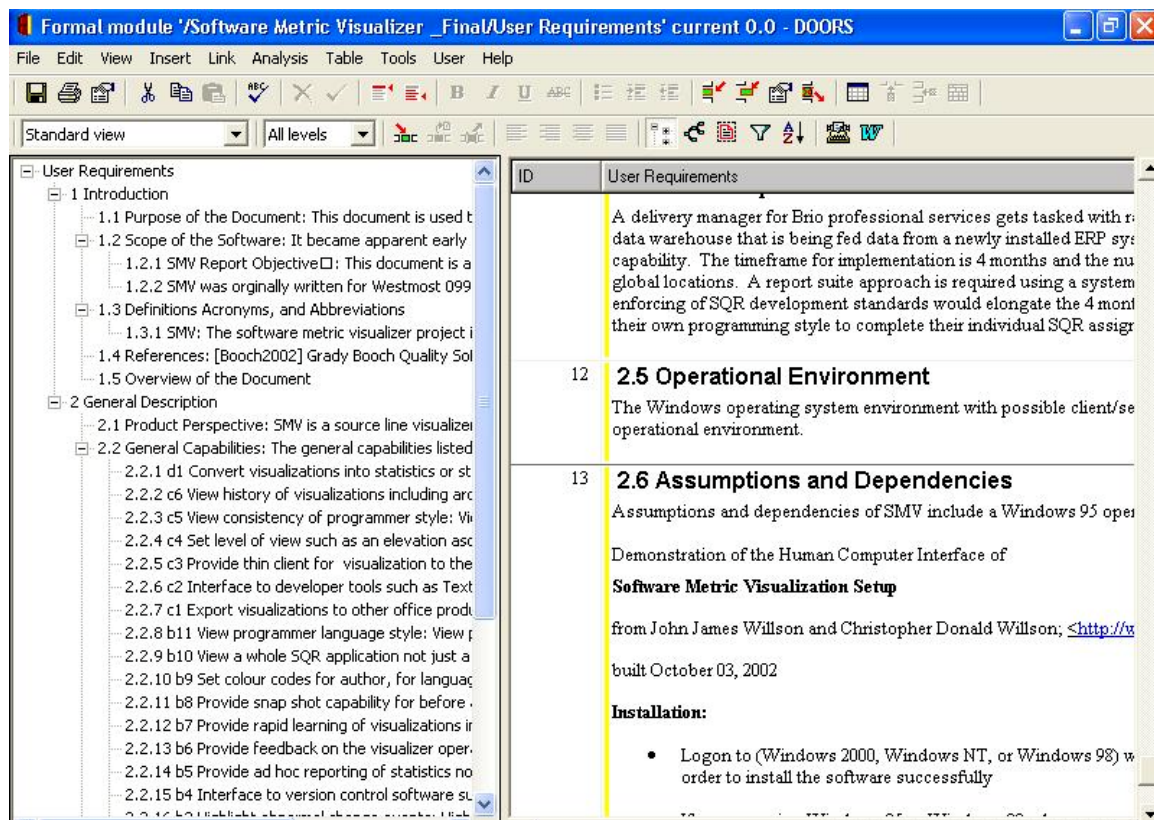
4.1 Conversion to Doors

Figure 4 Doors Directory Structure



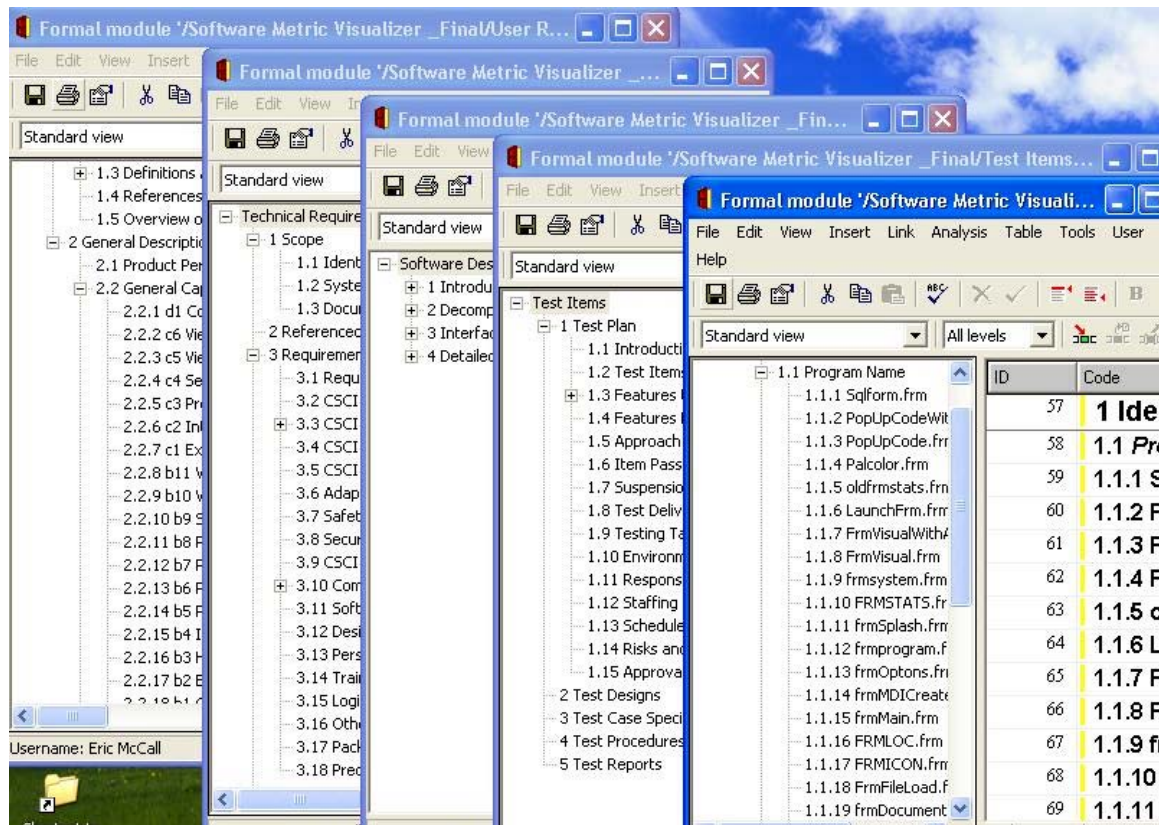
The Doors directory structure can model the Windows directory structure and this became an important way of visualizing SMV conversion to Doors. It also provided some ease of learning. Security prevented some forms of directory matching.

Figure 5 Example Start of 11 page Doors User Requirements Document



The above figure is provided to show the reader the capture of SMV requirements within the Doors product. Retention of the SMV requirements indexing is also shown in order to make the translation from before and after Doors. Further, the requirements indexed (2.2.1 d1 for example) were the first prototype requirements. In the second spiral of SMV development, end user tasking was used as high level requirements. Some features of the final SMV tool never made it to any of the three requirements' spirals. The use of a legend on each visualization of SMV just appeared as a nice-to-have feature and was never captured as a requirement.

Figure 6 Initial Doors SMV Documents

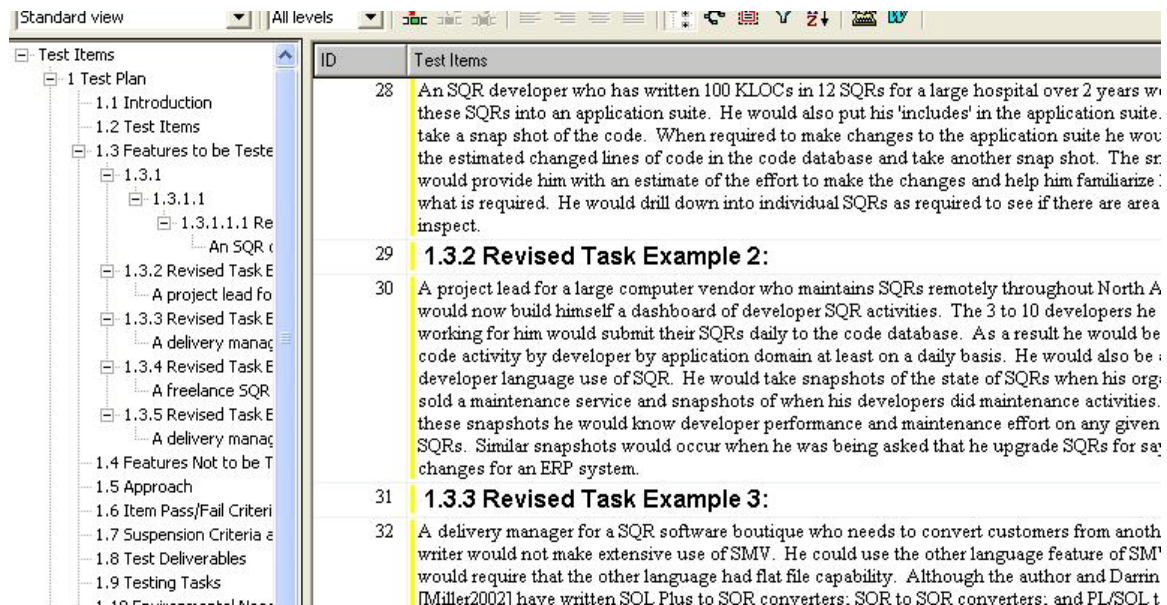


The above figure shows the many SMV documents and requirements that were captured and entered into Doors. The extreme left-hand document shows the initial requirements. The next pane shows technical requirements using a selected industry standard template. The middle pane shows attempts at graphic artifact requirements capture within the Doors product. The next pane shows testing and revision of some high level requirements to include end user task scenarios. These came directly from the user group prototype testing. The final pane shows the code developed and listed within the Doors product. Note the frm before the program name is the same frm of figure 3.

4.2 Identifying and Capturing Requirements

Using the work around previously discussed of exporting a style sheet and importing the revised style sheet, it became very easy to enter SMV requirements that had been previously typed. The ability to enter requirements directly within the Doors product was also performed and this too was quite easy. The figure below shows some of the end user task examples entered as requirements. These were originally not identified as requirements but through the use of Doors became easy to measure requirement benchmarks.

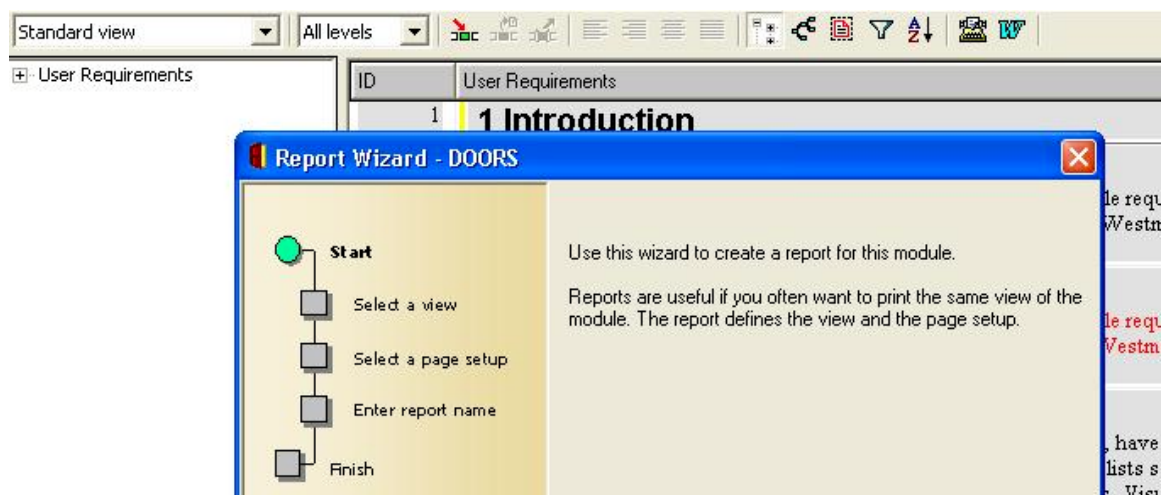
Figure 7 Doors Capturing of SMV Requirements



ID	Test Items
28	An SQR developer who has written 100 KLOCs in 12 SQRs for a large hospital over 2 years w these SQRs into an application suite. He would also put his 'includes' in the application suite. take a snap shot of the code. When required to make changes to the application suite he wou the estimated changed lines of code in the code database and take another snap shot. The sr would provide him with an estimate of the effort to make the changes and help him familiarize what is required. He would drill down into individual SQRs as required to see if there are area inspect.
29	1.3.2 Revised Task Example 2:
30	A project lead for a large computer vendor who maintains SQRs remotely throughout North A would now build himself a dashboard of developer SQR activities. The 3 to 10 developers he working for him would submit their SQRs daily to the code database. As a result he would be code activity by developer by application domain at least on a daily basis. He would also be developer language use of SQR. He would take snapshots of the state of SQRs when his org sold a maintenance service and snapshots of when his developers did maintenance activities. these snapshots he would know developer performance and maintenance effort on any given SQRs. Similar snapshots would occur when he was being asked that he upgrade SQRs for sa changes for an ERP system.
31	1.3.3 Revised Task Example 3:
32	A delivery manager for a SQR software boutique who needs to convert customers from anoth writer would not make extensive use of SMV. He could use the other language feature of SM would require that the other language had flat file capability. Although the author and Darrin Miller20021 have written SOL Plus to SOR converters: SOR to SOR converters: and PL/SOL t

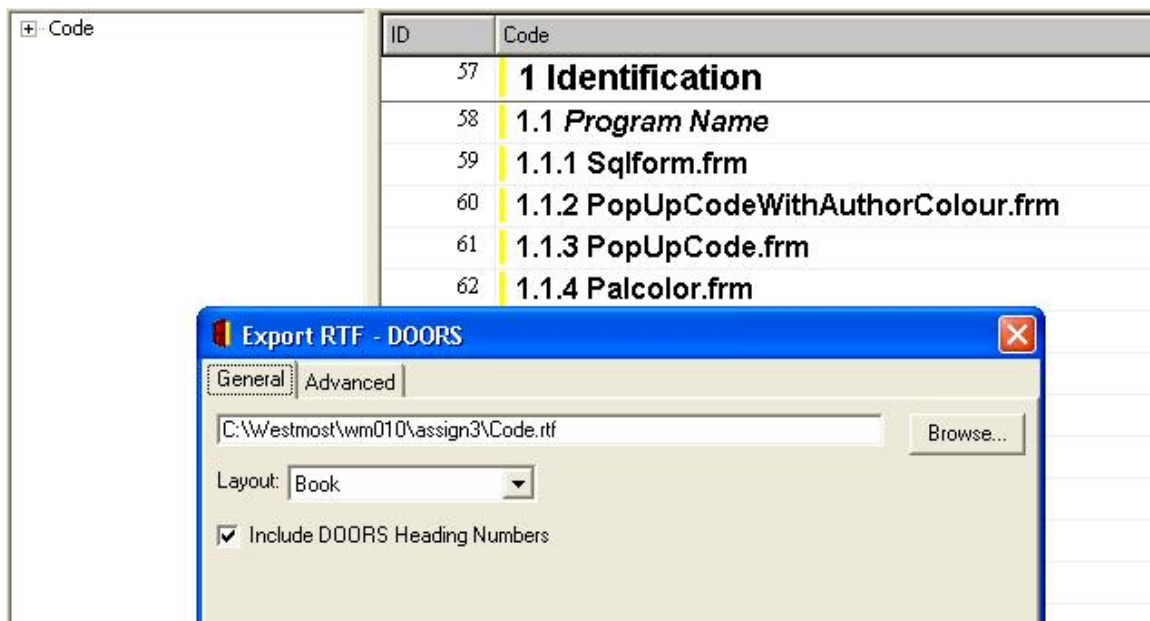
4.3 Features of the Doors tool used

Figure 8 Use of Doors Wizards



Not all features of the Doors tool were used. However, there were some notable features used. The previous figure shows the use of Doors wizards. The wizards provided considerable enhanced capability. Reports directly out of Doors could be produced. These reports in rich text format could be easily edited and highlighted with revised text. Originally it was envisioned that this report would be produced within Doors itself. Several problems with Doors handling of graphics prevented completing the report within the timeframe available. Exporting and importing of rich text formatted documents in multiple layouts was very easy as is shown below. All requirements could be captured in multiple ways and imported and/or exported as needed.

Figure 9 Use of Doors Import/Export of Rich Text



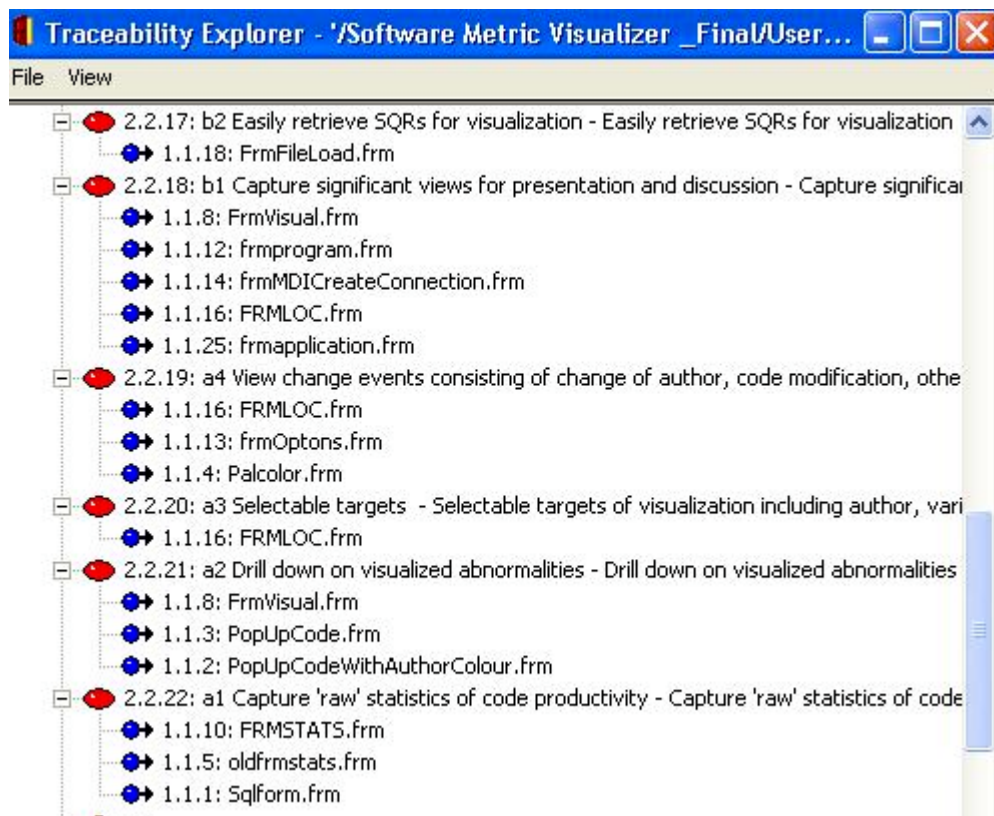
4.4 Requirements Project Flow

Doors enforced requirements project flow. Although the systems development life cycle of SMV was spiral, Doors had no problems capturing requirements in all 3 prototypes and in linking these requirements.

4.5 Tracing Requirements and Linking

One of the most exciting features of migrating SMV to Doors was the tracking, tracing, and linking of requirements. The figure below shows some of the links that were developed using the Doors product. This figure could be compared to figure 3 "Relationship of Requirements to Software Products without Doors". This is truly a way of providing insight into requirements engineering that would not exist with out some form of requirements management automation.

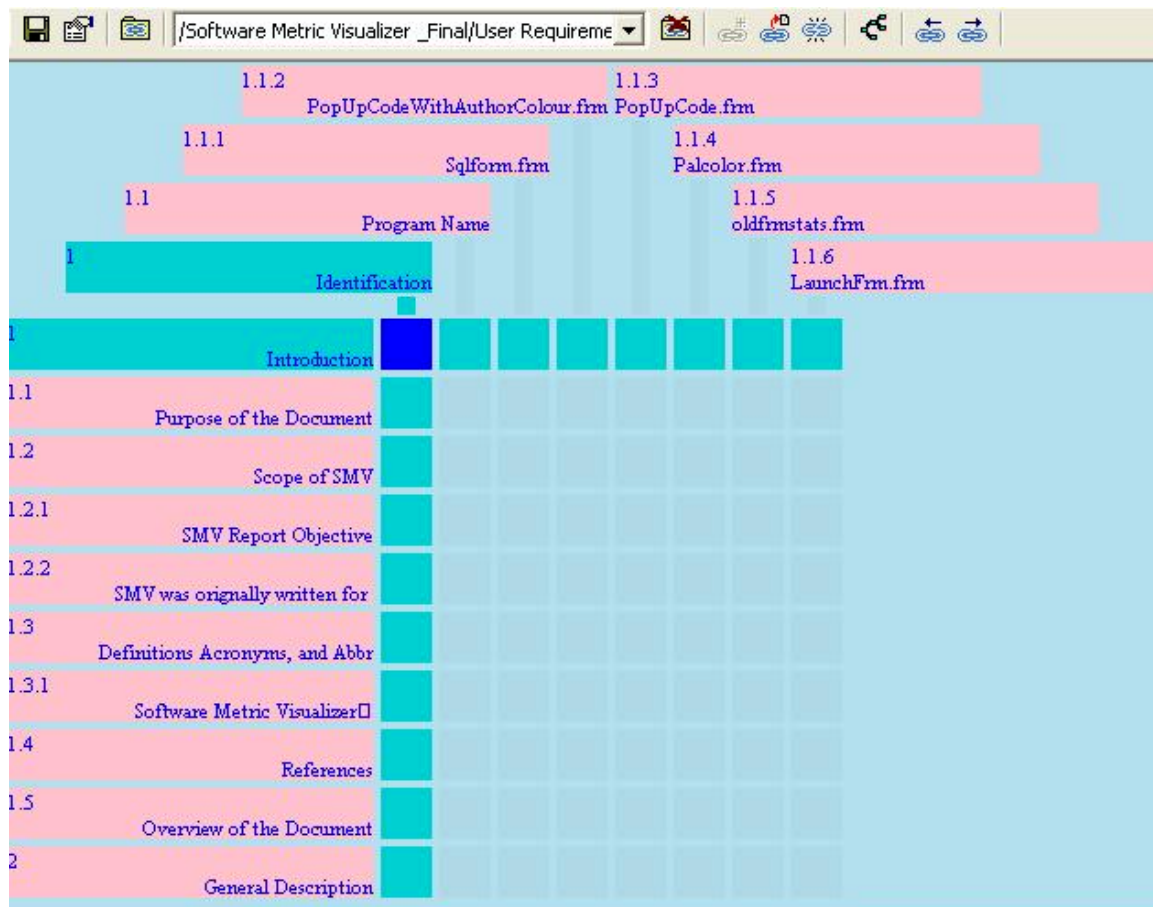
Figure 10 Doors Tracing Requirements through to Code



4.6 Verification of Requirement Completion

The requirements tracking would not be complete without analysis of how those requirements relate and interact together. Here Doors again provided insight that could only be accomplished prior to SMV being in Doors though extensive sweat equity. If the preliminary analysis of the requirements had been used through Doors, the spirals of SMV might have been better documented. For sure impacts of requirements on the spirals would have created more managed spiral development. Again, this was so readily easy to use that the value of a Doors implementation might be justified on this insight alone.

Figure 11 Use of Doors Analysis



4.7 Not Enough Low Level Requirements

It became clear when moving SMV requirements to Doors that many requirements were not fulfilled. But one of the more surprising results was that the original SMV requirements were not detailed enough. Although SMV was not completed using a project management package such as Microsoft Project, it was completed using milestone progress reports and Gantt charts. So higher level requirements were captured and to some extent managed. However; lower level requirements or next spiral requirement details were never fully captured, measured, or tracked. What a comment on the SMV project.

4.8 History of Requirement Changes Available

Since there were not enough low level requirements captured in SMV, this led to the questioning of whether there was adequate history of requirement changes in pre-Doors SMV. The answer is obvious. History of requirement changes was never documented. It was captured within the stakeholders and in particular by the user group testing and reviews. But it was never documented in even an

informal sense. Again, a clear demonstration of the advantages of a requirements management software tool. With respect to 'agile methods' requirements management, this is clearly not well done with a code-and-go approach either. [Eber2002]

4.9 DXL - Not for Everyone

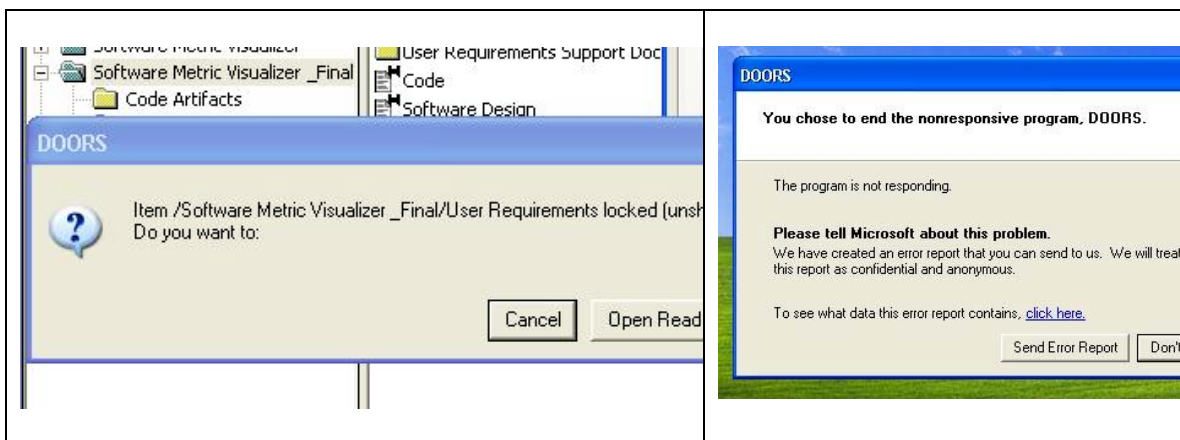
The conversion from SMV requirements to Doors did not require the use of DXL, the Doors programming language. As a result, the only comment made is that there may be features with direct applicability to SMV requirements that were never tried. Doors is quite feature rich.

4.10 Interfacing Out and Documentation

The following is a listing of whines and bouquets concerning the conversion of SMV to Doors and some of the problems encountered that may be addressed in other versions of Doors than the one provided for evaluation.

4.1 Microsoft Office Suite Products

Figure 12 Doors Bugs and Locks



Although mentioned earlier, the metaphors and related paradigms of Windows and Doors clash upon occasion. Doors does not gracefully fail but attempts to run a debugger that the end user has no access to. Further, the objects that Doors was working on at the time become locked by the parent object owner and appear to revert to earlier versions of the object. An alternative strategy would be to identify the location in the object that had focus at time of failure. Further, Doors objects do not retain sizing history of open views. They must be resized each time opened. Again insertion into the propriety database is very large system SQL like. This is not a problem once up the learning curve, but is counter to the Microsoft Office Suite look-and-feel. (Even though under the wraps Doors might be a Back Office application?) So a prioritized requirements list must be inserted in reverse order, in order to get correct numbering. Exporting to Microsoft Word did not work in Word format. However, it did work in rich text format.

4.2 Use the Web

Doors information through the use of the Telelogic resource centre and through straight web browsing was excellent. Relevant information could be obtained through white papers and user group productions.

4.3 Compiled Help

The compiled help feature of Doors was also excellent in providing understanding of the software. However, overview of requirements management could be better done and the why of requirements management could have been better done. Also, having to use the help indicated that the Doors product did not meet one of the earliest major expectations - " ease of use and rapid learning of a tool such as Doors with out the use of hardcopy or compiled help files and/or training."

5. CONCLUSIONS

The table below provides a summary of improvements by Doors / Non-Doors comparison on the SMV project. Note that not all features of Doors were used. Only those features that were needed for the SMV project.

Table 3 Summary Improvements With Doors

Comparison Description	Pre- Doors	Doors
Capture of Requirements	<ul style="list-style-type: none">- Humans had to convert requirements from interviews, questionnaires, JADS, etc.- Only office suite productivity automation	<ul style="list-style-type: none">- Humans had to convert requirements from interviews, questionnaires, JADS, etc.- In addition there are templates that represented industry association standards- Doors had a decided edge in ease of subsequent use as is shown in this document
Tracing Requirements	<ul style="list-style-type: none">- Prior to seeing Doors the only tracing of requirements was end user testing and end user memory	<ul style="list-style-type: none">- Doors is absolutely better in that the linking of requirements through multiple project artifacts was point and click.
Changing Requirements and Impacts	<ul style="list-style-type: none">- Requires intimate knowledge of all system development life cycle products including the code.	<ul style="list-style-type: none">- Because of the trace linking impact assessment is a matter of following the linking and is very easily done.
Maintenance	<ul style="list-style-type: none">- Market or user driven, not	<ul style="list-style-type: none">- Easy to find relationships among

of SMV	necessarily based on need.	requirements. - Major revisions are subsequently possible with their impacts identified
Focused Development Effort	- Based on end user or developer input	- Objectively based on agreed prioritization. - Development effort is scalable with easier to identify budget high water marks.
Assurance all requirements met	- In relating requirements to code it is obvious there are missing requirements and code not matched to requirements.	- All requirements were linked to code and 'orphaned' code was clearly shown.
Index of Requirements	- Index and numbering of requirements was manual and based on a 'home grown' system.	- Indexing was automatic. For Doors testing purposes both numbering approaches were used and this showed the problems of a manual approach.
Linking of Requirements	- Never done in original project. - Tedious after the fact activity	- Drag and drop linking!
Detection of Requirements Interaction	- Not readily possible. Relies on intuition of developer.	- Absolutely clear, recordable, and actionable.

5.2 Interface Improvements

It is clear from this report that some approach to importing and exporting should be rethought within Doors for the novice or inexperienced user. A possible option here might be to extend the excellent use of Doors wizards to check for import/export document styles or templates. Since this is such an easy idea it is likely available with an alternative version of Doors or from a third party after market vendor.

5.3 Excellent Collaboration

Although not tried with SMV, the collaboration capability of Doors seems excellent. Most developers would like to know their contribution to a given project. Having requirement fan-in and fan-out complete with requirement

chaining, developers would be able to view their code production to requirements met. Quite a concept for some low level CMM shops.

5.4 Change the Doors Metaphor

The Doors metaphor to the Windows environment is excellent. However, it needs to be extended with other Microsoft Office Suite look-and-feel. Many customers of the Doors customer base might be using alternative platforms including alternative office suites; however, if rich text formatting is the interchange media, then Doors should present it that way to the novice or intermediate user. The spread sheet metaphor fits with many technocratic views of requirements engineering; however, much progress in requirements engineering occurs through the use of graphics, diagrammatic, and craftsman like approaches to doing requirements management.

5.5 Overhead versus Efficiency

The author's original concerns in using a product such as Doors to do requirements management as being overhead intensive with out any pay back were wrong. Using SMV with Doors it became very clear that the comparison of pre-Doors and post-Doors requirements was not a fair comparison. Doors provides a level of insight that is not readily apparent in the crush of doing projects. This insight could translate into better products and more timely projects. There is a role to fill by requirements management automation tools that has applicability even on small projects.

5.6 Putting this Document in Doors

However, the Willson's test of using Doors as a requirements management tool for all projects, requires that this document be readily imported into Doors. And that during that importing that the requirements are extracted from this document for the son of SMV project. Further the graphics and tables used be imported also.

Appendices

A. References (This document)

[Eber2002] Armin Eberlein & Julio do Prado Leite,
Agile Requirements Definition: A View from Requirements Engineering,
<http://www.enel.ucalgary.ca/People/eberlein/publications/index.html>

[KotoSom1998] Gerald Kotonya & Ian Somerville, *Requirements Engineering: Processes and Techniques*, John Wiley & Sons Ltd, West Sussex, England

[Tele2003] <http://www.telelogic.com/>

B. References (SMV exported from Doors)

Pasted from exported Doors User Requirements Document

"1.4 References"

[Booch2002] Grady Booch *Quality Software and the Unified Modeling Language*,
Rational Software Corporation, 2002, <http://www.rational.com/>

[Brio2002] *Brio Performance Suite 8*, Brio Software <http://www.brio.com/>

[Burton1994] Peter Burton *SQR User's Guide and Developer's Kit* 1994 MITI Long
Beach, CA

[Dask1992] Michael K. Daskalantonakis *A Practical View of Software Measurement and Implementation Experiences Within Motorola* IEEE Transactions on Software Engineering, vol. 18, no. 11 (November 1992), pp. 998-1010.

[Dbminer2002] DBMiner, DBMiner SX 2002, <http://www.dbminer.com/>

[Dumke1999] R. Dumke *Metrics Tools - An Overview*. Metrics News, 4(1999)1, pp. 21-28

[Eick1992] S.G. Eick; Steffen, J.L.; Sommer, E.E.: *Seesoft -- A Tool For Visualizing Line Oriented Software Statistics*. IEEE Transactions on Software Engineering, 18(1992)11, pp. 957-968

[Eick1996] Thomas A. Ball and Stephen G. Eick. *Software visualization in the large*, IEEE Computer, April 1996, pp. 33-43.

[Emden2002] Eva van Emden, Leon Moonen, jCosmo - *Java Code Smell Browser Tool*, [<http://www.cwi.nl/projects/renovate/javaQA/>](http://www.cwi.nl/projects/renovate/javaQA/)

[Florac1999] William Florac and Anita, *Carleton, Measuring the Software Process*, Addison Wesley, Reading Mass.

[GM2002] FQS Poland, *User Manual*, Ghost Miner, 2002,

[Gutwin1999] Carl Gutwin, *Visualizations of Interaction*, Technical Report 99-1, HCI Lab, University of Saskatchewan, 1999

[Knight2000] Claire Knight, *System and Software Visualizations*, Handbook of Software Engineering and Knowledge Engineering, 2000, [<http://www.durham.ac.uk/>](http://www.durham.ac.uk/)

[Kuh1994] I. Kuhrau *A Tool-Based Analysis of Borland C++* Master's Thesis, University of Magdeburg, February 1994.

[Landres1999] Galina Landres and Vlad Landres *SQR in Peoplesoft and Other Applications* 1999 Manning Publications, Greenwich, CT

[Miller2002] Darrin Miller *Harnessing SQR* Brio Software User's Conference November 2002, [<http://www.brio.com/>](http://www.brio.com/)

[Mellen1998] Don Mellen *SQR Programmer Reference 1998*, Ray Ontko & Co, Richmond, Indiana

[Norman1990] Donald Norman, *The Design of Everyday Things*, Doubleday, New York, New York

[Nielsen1999] Jakob Nielsen *Interfacing with Jakob Nielsen* June 1999 [<http://www.useit.com/>](http://www.useit.com/)

[Qsm2002] QSM Company, *The SLIM Software Tool Suite*, 2002, [<http://www.qsm.com/>](http://www.qsm.com/)

[Sgi2002] Silicon Graphics Mindset , 2002, [<http://www.sgi.com/go/mindset/>](http://www.sgi.com/go/mindset/)

[Sorenson1993] Boloix, G., Sorenson, P.G. and Tremblay, J.P Software "Metrics using a Metasystem Approach to Software Development", *International Journal of Systems and Software* 20, 1993: 273-294.

[Swanson2001] Gregory Swanson and Lee Globus *Visualization for a Graphical Programming Environment*, Nielsen 2001, [<http://www.tslice.com/>](http://www.tslice.com/)

[Tao1999] Xie Tao, Huang Huang, Xiangkui Chen *Object Oriented Software Metrics Technology* 1999 Ricoh Company Ltd. Tokyo, Japan & Software Quality Evaluation Group Peking University

[Toth1996] R. John, J. Madhur, R. Stewart, K. Toth, *Software Quality Metrics Process For Large Scale Systems Development*, 1996 INCOSE Symposium, July 1996

[Willson1998] John Willson, Stephanie Crafford *U.S.A. Computer Consulting - For Canadians and Other Aliens*, 1998, DSS Ltd. [<http://www.dssltd.com/>](http://www.dssltd.com/)

[Willson2001] John Willson *EXtreme SQR Programming*, Unpublished Brio 2001 presentation, [<http://www.dssltd.com/whitepapers/>](http://www.dssltd.com/whitepapers/)

[Wong1996] Kenny Wong *On Inserting Program Understanding Technology into the Software Change Process* Fourth Workshop on Program Comprehension (WPC 1996)

[Wong1999] Kenny Wong *The Reverse Engineering Notebook*, PhD. Thesis, Department of Computer Science, University of Victoria [<http://www.cs.uvic.ca/>](http://www.cs.uvic.ca/)