Interoperability
Between
The DOORS Requirements Management Tool
And
The CORE Systems Engineering Tool

Jody H. Fluhr
Senior Systems Engineer
Vitech Corporation
www.vitechcorp.com

Pat Macdonald
Senior Consulting Engineer
Vitech Corporation
www.vitechcorp.com

Many organizations use tools such as DOORS to facilitate basic requirements and document management. By leveraging the interoperability with CORE, projects extend automated support to the entire systems engineering effort including analysis, architectural, and management aspects critical to project success. This paper outlines the commonality and contrasts between the CORE and DOORS tools that affect data exchange, including the terminology and structure of each tool's constructs. A mapping between each tool's terminology/constructs is provided as well as a step-by-step illustration of one approach to port data between the tools.

DOORS™ Overview

DOORS by Telelogic, Inc. is a requirements management application. It provides features to capture, track and manage user requirements. Requirements can be directly entered into DOORS, using its familiar word processor style interface. Requirements can also be imported into DOORS from a wide variety of file formats such as Microsoft Word, Excel, PowerPoint and Outlook, HTML, plain text, etc. DOORS provides features to develop views of requirements, links between requirements and documents, and reports and supports traceability analyses. DOORS also provides a versioning capability to track changes to data and provides a change management capability to facilitate change processing tasks. DOORS runs under the Microsoft Windows® environment in a workstation or networked configuration and on UNIX platforms.

CORE® Overview

CORE, a product of Vitech Corporation, is a systems engineering CASE tool designed to facilitate the systems engineering process in any application. CORE provides features to manage requirements, develop functional (behavioral) and physical (architectural) models, and develop and execute discrete-event simulations. CORE is based on a powerful object-oriented database and a modifiable schema. It generates numerous diagrams based on data in the database such as Enhanced Functional Flow Block Diagrams (EFFBDs), hierarchy/traceability diagrams, physical interface diagrams, IDEF0 diagrams, and N^2 diagrams. CORE provides the ability to construct queries and reports that can be utilized to generate system documentation such as specifications, interface control documents and traceability reports. CORE runs under the Microsoft Windows environment in a workstation or networked configuration.

Interfacing CORE and DOORS – Terminology/Semantic Issues

Both CORE and DOORS have a database as a foundational element of their architecture. Each tool's database has its own schema, user interface, and routines that allow the user to enter, modify, and process data contained in the database. Both tools provide capabilities to allow users to modify and extend the schemas, capabilities that are useful in tuning the tools to operate in tandem.

The similarities between the CORE and DOORS databases make interoperability between the tools possible. Constructs in the DOORS database map readily to CORE constructs as shown in Figure 1.

DOORS Constructs	CORE Constructs
Formal Modules	Classes
Objects	Elements
Object Attributes	Element Attributes
Object Links	Element Relationships

Figure 1 - A Mapping between DOORS and CORE

Figure 2 through Figure 4 show how the different constructs map in context of each tool's graphical user interface.

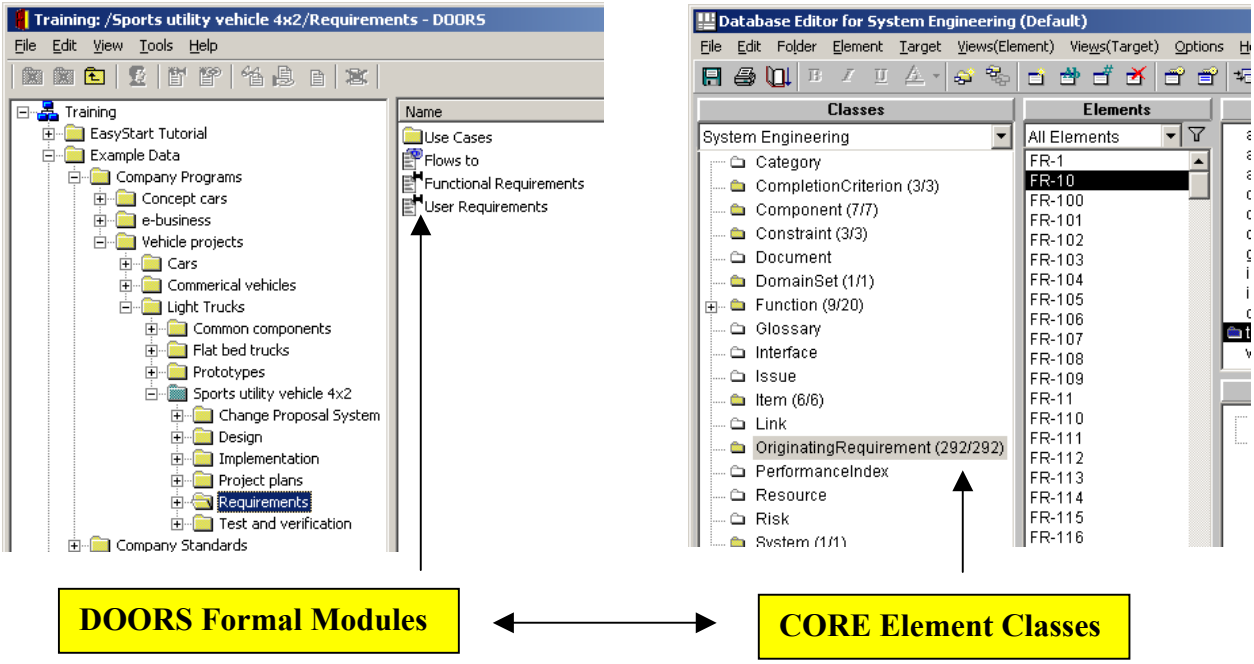

Figure 2 – DOORS Formal Modules and CORE Element Classes

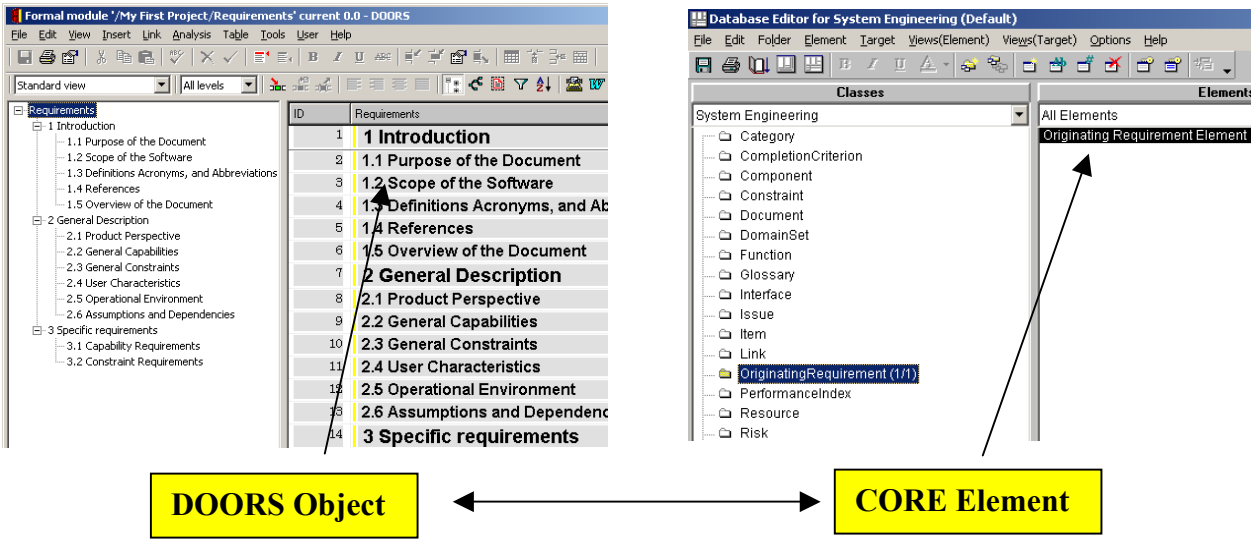

Figure 3 – DOORS Objects and CORE Elements

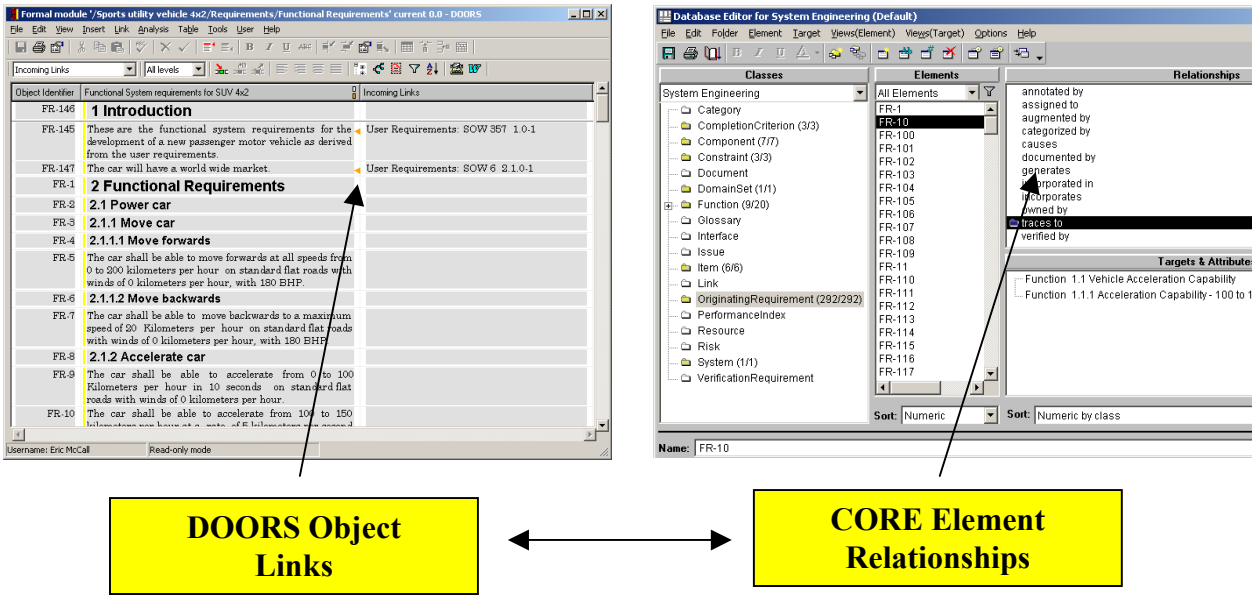

Figure 4 – DOORS Links and CORE Relationships

Interfacing CORE and DOORS – Syntactical Issues and Mechanisms

CORE has a strongly typed schema – specific classes of elements and relationships between those classes of elements are defined and enforced. In terms used in Object-Oriented Design (OOD), "classes" in CORE are synonymous with "classes" of OOD; "elements" are instantiations of CORE classes much like "objects" are instantiations of classes in OOD. The relationships between classes, establishing their roles in analysis and management, are also defined. Relationships are strongly typed in that every relationship type has a specific name and is defined between specific classes of elements. What differentiates one class from another is the attributes that a given class element can have and the relationships a given class can have with other elements. Every instantiation of an element of a class has

the same attribute fields and potential relationships as other element instantiations of that class. In CORE 4.0, a database element is distinguished from all other elements in the database by its class and its name (in later versions, a unique identifier is automatically assigned by CORE).

DOORS provides a loosely typed schema. In DOORS, Formal Modules are created and defined by the user to contain objects with the same attribute fields. In terms used in object-oriented design, "formal modules" in DOORS are synonymous with "classes" of OOD; "objects" are instantiations in formal modules classes much like "objects" are instantiations of classes in OOD. In DOORS, an object in the database is distinguished from all other objects in a formal module by an object identifier, which is automatically assigned by DOORS.

As previously stated, "relationships" in CORE are strongly typed or well defined. The standard CORE schema establishes specific relationships from one class to another. One feature of CORE is that all relationships are coupled to form a bi-directional binding between elements. For instance, a relationship labeled, "traces to" is a common relationship used to show how an element in the "OriginatingRequirements" class "traces to" an element in the "Function" class. When the user creates this relationship, the reverse direction "traced from" shows the relationship from the perspective of the "Function" element. Therefore, relationships are single objects that are bi-directional in nature. This feature provides the forward-backward or up-down traceability between elements in the CORE database.

In DOORS, relationships are referred to as "links". Links can be established between any two objects within the database and do not have specific names and roles as in CORE. Links are directional in nature, and unlike CORE, DOORS requires the user to establish links between objects in both directions to establish traceability in both directions.

Process Description

There are numerous ways to establish interoperability between CORE and DOORS. Solutions range from simple manual procedures to highly automated processes. The ways an organization wants to utilize both tools on a project will determine how constructs and information are mapped. For the sake of understanding how to port data from one tool to the other, consider the following mapping representing a simple usage scenario in which DOORS manages the initial system requirements and CORE supports the engineering and analysis efforts:

1. Formal modules in DOORS associated with requirements will be mapped to elements in the CORE *OriginatingRequirements* class.

2. The DOORS *absolute number* attribute will serve as the CORE element *number* attribute. Each element created in CORE will have a unique number and will serve as the absolute number for those elements ported back to DOORS.

3. Data from the various CORE classes (*Functions*, *Constraints*, *Interfaces*, etc.) will be mapped to formal modules in DOORS defined as Functions, Constraints, Interfaces, etc.

4. Links between objects in DOORS will be mapped to relationships between elements in CORE. For relationships formed within CORE, DOORS' "link by attribute" capability will be utilized to port those relationships into DOORS.
    a. Since the "link by attribute" feature is to be used in DOORS, the unique object identifier must be known prior to data export from CORE. CORE must assign the unique number.
    b. To assign the unique number, a script is executed within CORE that will assure that each object to be exported has a unique number.
    c. For every relationship formed in CORE that is to be ported to DOORS, a *"Link by Attribute"* field will be exported.
    d. Once the data has been imported into DOORS, the links between the formal module objects must be established. This is accomplished by utilizing the "Link by Attribute" capability in the "Link" pull-down menu item.

An Illustration

Consider a simple project with two requirements. Figure 5 shows the requirements in DOORS.

Both CORE and DOORS have the capability to import and export data in a comma-delimited file format. This is the easiest method to use since both tools provide the capabilities without requiring the development of customized scripts. This method is illustrated below.

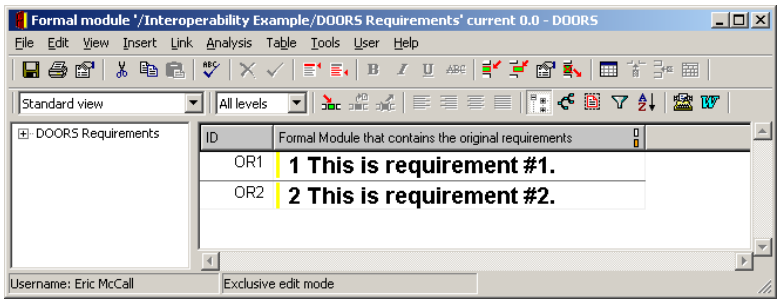

Figure 5 – DOORS Formal Requirements Module

Porting Data from DOORS to CORE

1. To begin the process, a formal module is opened in DOORS. From the file menu, the Export to Spreadsheet selection is made as shown in Figure 6.
2. Since it may not be desirable to export all the attributes of the objects in the formal module, the capability to select attributes from a list is exercised. This is performed by selecting the "Attributes from list" option in the dialog box as shown in Figure 6. For this example, only the "Object Identifier", "Absolute Number" and the "Object Heading" are selected.
3. Once the attributes have been selected and a file name is specified, the "Export" button is depressed to produce the comma-delimited file. The resulting Microsoft Excel file is shown in Figure 7. Note that the column headings in the file reflect the attribute names.
4. To map the data from the DOORS database to the CORE database, some minor modifications to the comma-delimited file are required. (For a defined usage scenario, these transformations could be avoided by using a customized version of the generic import script.)
    a. The "Object Identifier" column heading is changed to "Name", which is the attribute recognized by CORE to uniquely identify an object.
    b. The "Unique Number" is changed to "number".
    c. The "Object Heading" column heading is changed to "description".
    d. An additional column with a "Class" heading is added to specify the class of element to be created using the data exported from DOORS. The resulting file is shown in Figure 8.
5. The modified comma-delimited file is imported into CORE using a standard parser script supplied with the tool. The script processes the file to produce a CORE-compatible database import file. This is accomplished by selecting "Scripts" on the main menu bar of the CORE control panel and selecting "Run Script" as shown in Figure 9. The "Basic CSV File Parser" script is selected from the list of scripts provided with CORE. A prompt allows the user to specify the file name and path for the script output that is the CORE database import file.
6. The database import file is imported into CORE using the "Import Database" button on CORE's control panel as shown in Figure 10.
7. CORE imports the data and creates the elements in the appropriate class. The populated CORE database is shown in Figure 11.

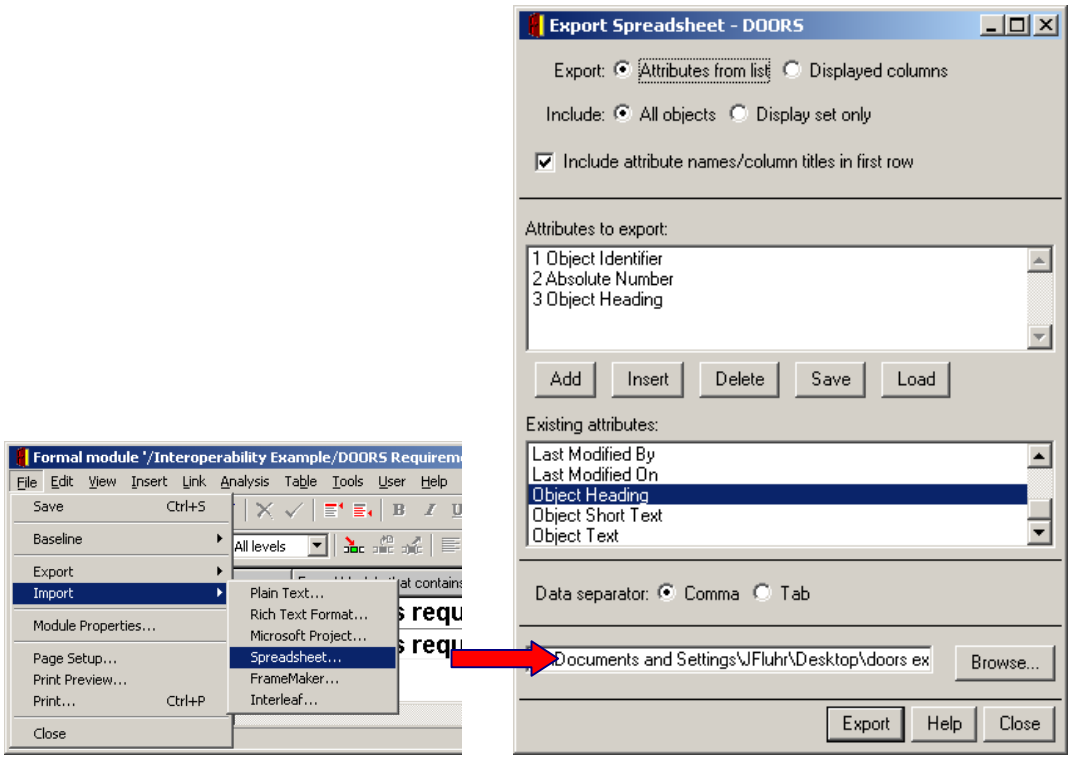

Figure 6 – Exporting Requirements Modules to Spreadsheets

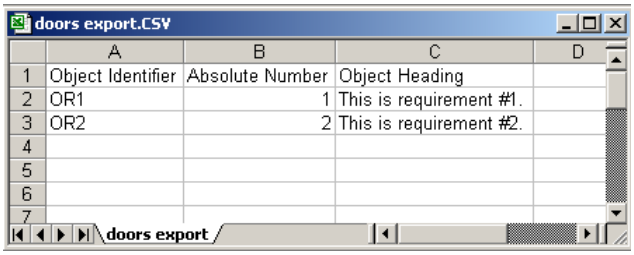

Figure 7 – Resultant Spreadsheet Output

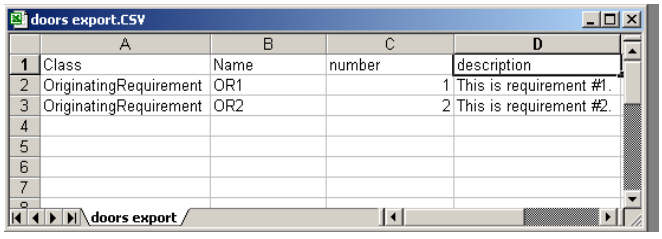

Figure 8 – Modified Spreadsheet (Note New Column and Column Heading Changes)

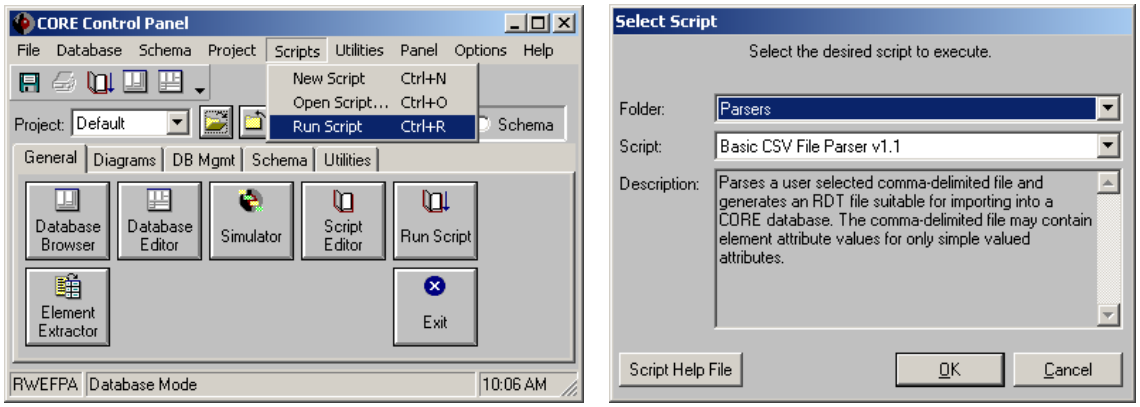

Figure 9 – Starting the Spreadsheet Import Process

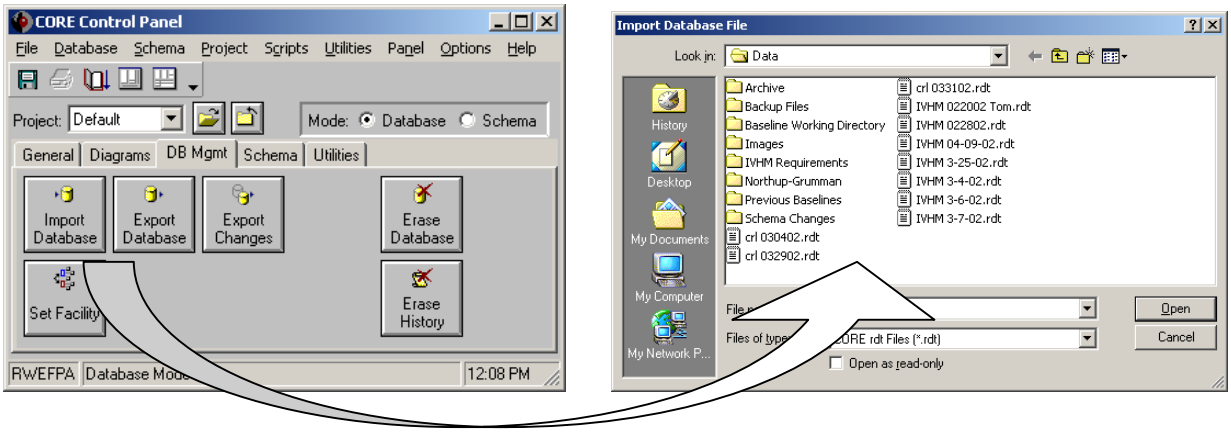

Figure 10 – Importing the Database File (Generated by Script)

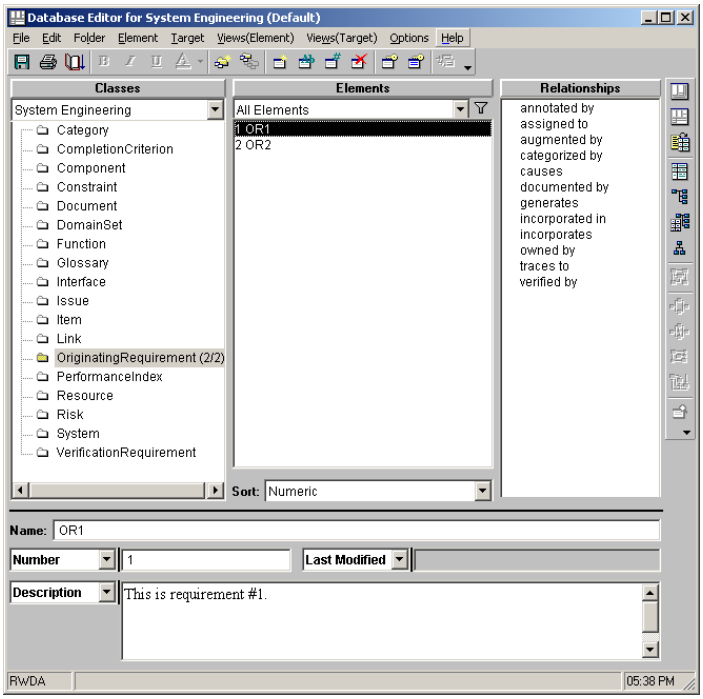

Figure 11 – Imported Requirements in CORE (Originating Requirements)

Continuing the Illustration – Moving Data from CORE to DOORS

For this example, two elements – derived from the two requirements ported from DOORS – are created in CORE: a component (component_001) and a function (function_001) as shown in Figure 12.

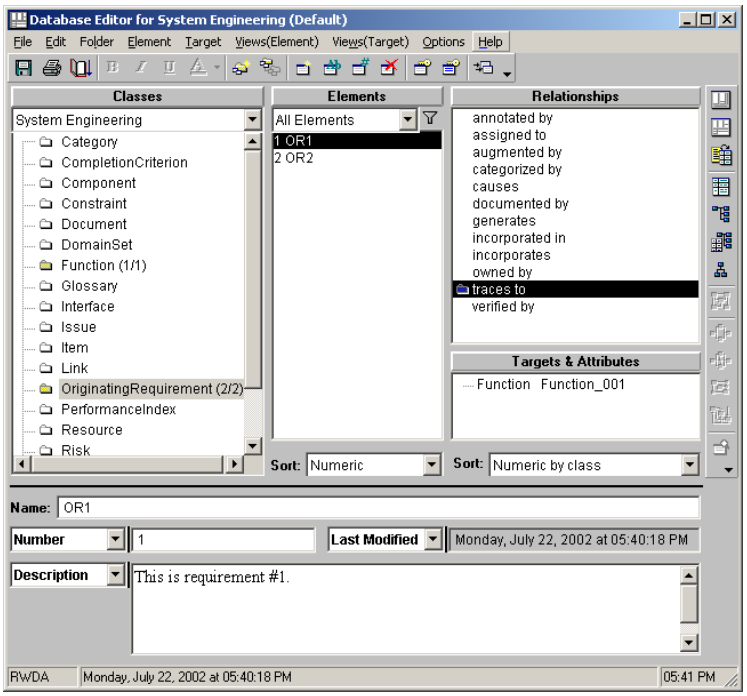

Figure 12 – Tracing Requirements to Functions and Components

These new CORE elements and their relationships to the requirements can be ported back to DOORS.

Porting Data from CORE to DOORS

1. To export data from CORE to DOORS, a script provided with the tool is utilized to produce a comma-delimited file. Since classes in CORE map readily to formal modules in DOORS, a comma-delimited file for each class of elements in CORE, such as Functions, Constraints, Interfaces, etc., is created to import to formal modules in DOORS. This is accomplished by selecting “Scripts” from the CORE Control Panel main menu and selecting “Run Script” as shown in Figure 13.
2. The “Generate CSV File” script is executed which will prompt the user for class, elements, attributes, and relationships to export. For this example, only the element name, description, and “traced from” relationship is exported. This process is performed for the *Function* class element and then the *Component* class element. The resulting comma-delimited files are shown in Figure 14.
3. To map the CORE data elements to DOORS objects, the spreadsheet headings are modified:
    a. The “Class” column is deleted.
    b. The “number” column is renamed to “Absolute Number” and is moved to column 1.
    c. The “Name” column is renamed to “Object Heading”.
    d. The “Folder” column is deleted.
    e. The “description” column is renamed to “Object Text”.
    f. The “traced from” column is renamed to “Traced From”.

4. In DOORS, Formal Modules are created for each class of data exported from CORE. Once the formal modules are created, the comma-delimited files can be imported. The resulting formal modules are shown in Figure 16.

5. To establish links that reflect the relationships established within CORE, the user utilizes the "Link By Attribute" functionality in DOORS. The resulting requirements, function, and component formal modules are shown in Figure 17.

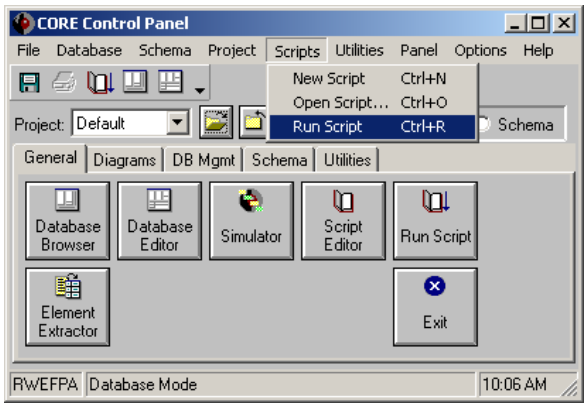

Figure 13 – Starting the Spreadsheet Export Process

core functions.csv

	A	B	C	D	E	F
1	Class	Name	Folder	number	description	traced from (OriginatingRequirement)
2	Function	Function_001	Function	1	This function was derived from requirement #1.	1
3						

core components.csv

	A	B	C	D	E	F
1	Class	Name	Folder	number	description	traced from (OriginatingRequirement)
2	Component	Component_001	Component	2	This component was derived from requirement #2.	2
3						

Figure 14 – The Resulting Comma-Delimited Files

core functions modified.csv

	A	B	C	D
1	Absolute Number	Object Heading	Object Text	Traced From
2	1	Function_001	This function was derived from requirement #1.	1
3				

core components modified.csv

	A	B	C	D
1	Absolute Number	Object Heading	Object Text	Traced From
2	1	Component_001	This component was derived from requirement #2.	2
3				

Figure 15 – Modified Spreadsheets

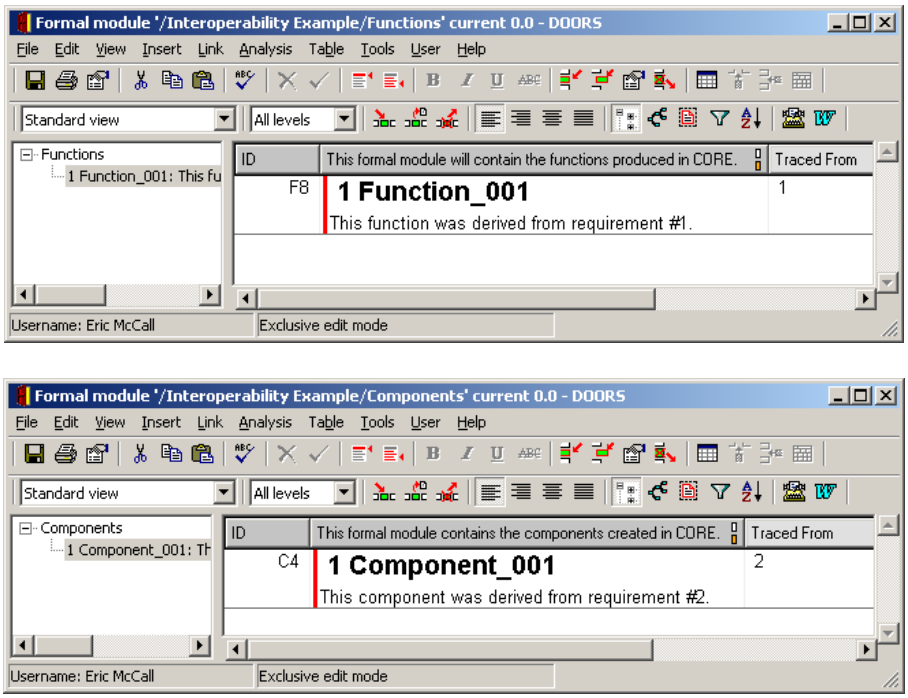

Figure 16 – The Resulting Formal Modules without Links

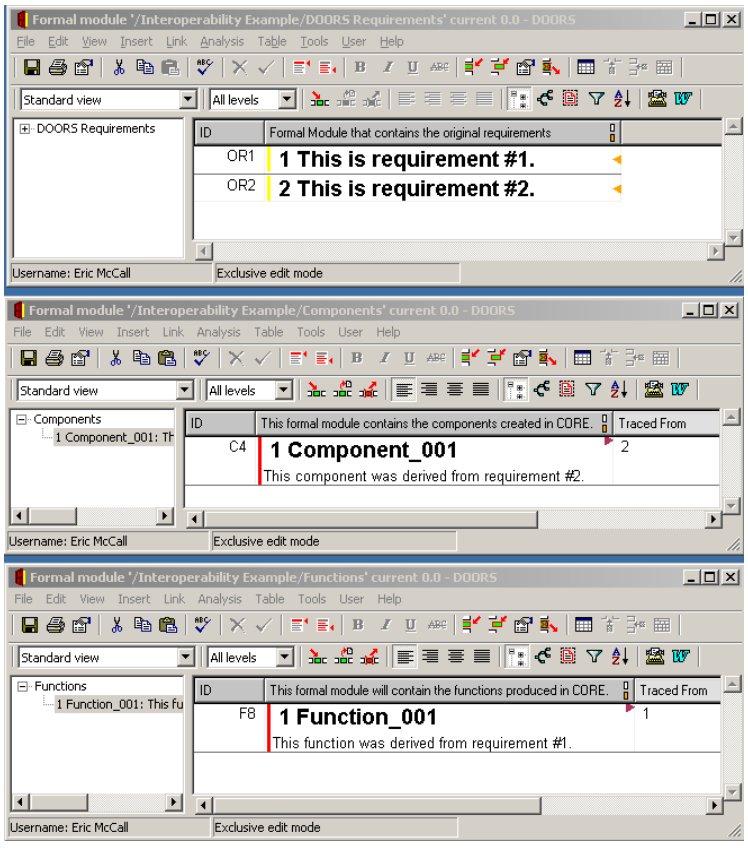

Figure 17 – The Resulting DOORS Formal Modules with Links

Conclusions

DOORS and CORE are compatible tools capable of exchanging information as part of an organization's integrated systems engineering environment. Once the terminology and constructs of both tools are understood, creating the mapping to direct data exchange between the tools is easily developed. As demonstrated above, a solution to port data between the tools is readily available and easy to implement.

The discussion above presents the details of one mechanism to port data between DOORS and CORE. Other more elaborate methods of data exchange are possible since each tool provides an Application Programmers Interface (API) as well as scripting languages that are used within each tool's environment. For instance, the necessity of modifying the comma-delimited output of each tool prior to importing the data to the other tool can be eliminated by developing custom scripts that properly format and label the data in the resulting output file. It is conceivable that a solution can be developed that allows data exchange between two C/C++ applications that utilize each tool's API without requiring manual operations. The solution your organization adopts should be tailored to meet the requirements and use cases that reflect your engineering processes.

Further Information

For more information about this topic, the CORE systems engineering tool, or other Vitech Corporation products and services, contact the Vitech Corporation at (703) 883-2270 or visit our website www.vitechcorp.com.

Updated July 2002