Release

3.1

CORE
A Guided Tour

CORE®: Product & Process Engineering Solutions

Restricted Rights Legend

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

Vitech Corporation
2070 Chain Bridge Road, Suite 320
Vienna, Virginia 22182-2536

Last revision date: December 2000

Table of Contents

Introduction

Thank you for your interest in CORE - *the* product and process engineering tool for the PC. CORE is an affordable tool that runs under the Windows OS to support system engineers during the initial phases of system definition, analysis, and design. Using the Windows interface enables CORE to be easier to learn and easier to use than its predecessors. The user-friendly interface allows efficient manipulation and representation of the system definition data. These interfaces include a *database browser*, a *system database editor*, an *element extractor*, and *engineering views - text*, *hierarchy*, *entity-relationship* (*ER*), *entity-relationship-attribute* (*ERA*), *function flow block diagram* (*FFBD*), *enhanced function flow block diagram* (*EFFBD*), and *N2* (*interface*) *charts*.

- This trial version* of **CORE** includes a sample database, **Collection Management System**, as it has been captured in the tool. The sample solution is presented in the data (ASCII) file: *CollMgt.rdo* in the SAMPLES directory.
- You can use this guide to recreate the solution yourself from scratch as we go along (starting on page 10) in **CORE**.
- This example is not intended to demonstrate the full power and flexibility of **CORE**. Instead, it serves as a simple, *structured walkthrough* of a sample product and process engineering problem in order to introduce you to the basic concepts and capabilities of **CORE**.
- In addition to top-down system engineering, **CORE** supports reverse (bottom up) and middle-out (incorporating legacy elements) engineering processes/paradigms. Some of **CORE's** most noteworthy applications have involved middle-out and reverse engineering solutions.
- While **CORE** supports the complete system engineering process in both *model-driven* and *document-driven environments*, relaxing the behavior modeling (from executable down to just hierarchies) provides a very powerful *Requirements Management* (*RM*) tool.
- To get a quick, but comprehensive, understanding of **CORE**, Vitech recommends our three-day introductory class. Please contact Vitech for more information on our training classes.
- We will be more than happy to provide additional information regarding our quick start and evaluation versions of **CORE**, product demonstrations, and our classes on product and process engineering and the application of **CORE**.

Vitech Corporation

2070 Chain Bridge Road, Suite 320
Vienna, Virginia 22182-2536
(703) 883-2270 FAX: (703) 883-1860
Email: info@vtcorp.com
Support: support@vtcorp.com
WWW: http://www.vtcorp.com

*The trial version is a limited 3.0 version of CORE. The current distributed version offers many enhancements and features not demonstrated in this manual.

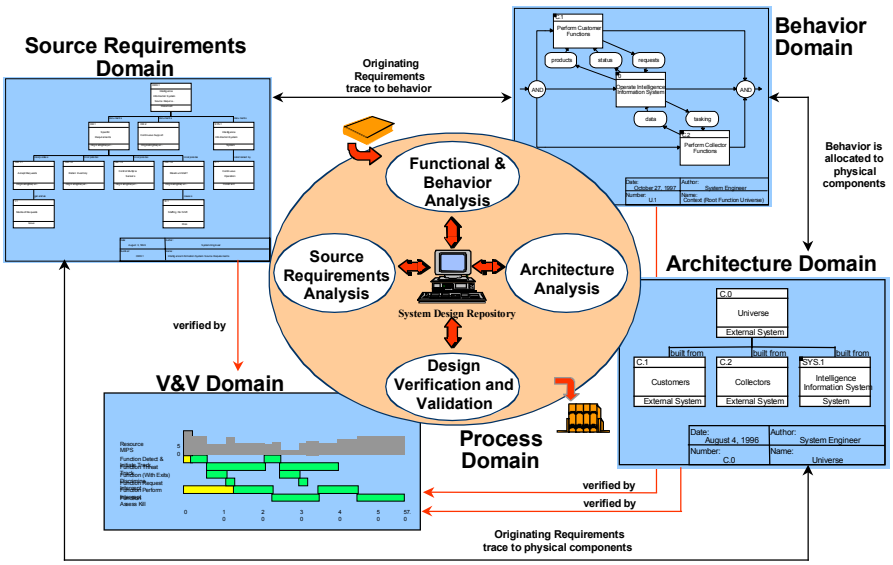
This Page Intentionally Left Blank.

Examining CORE

In this section, you get a feel for the CORE Tool

- **See how CORE applies to System Engineering**
- **Launch CORE**
- **Look at basic CORE Menus and Windows**
- **Import a data file into CORE Trial**
- **Export (Save) a data file from CORE Trial**

CORE: The Complete Process



The diagram above represents the **CORE** paradigm for product and process engineering. It serves as a foundation for our walkthrough.

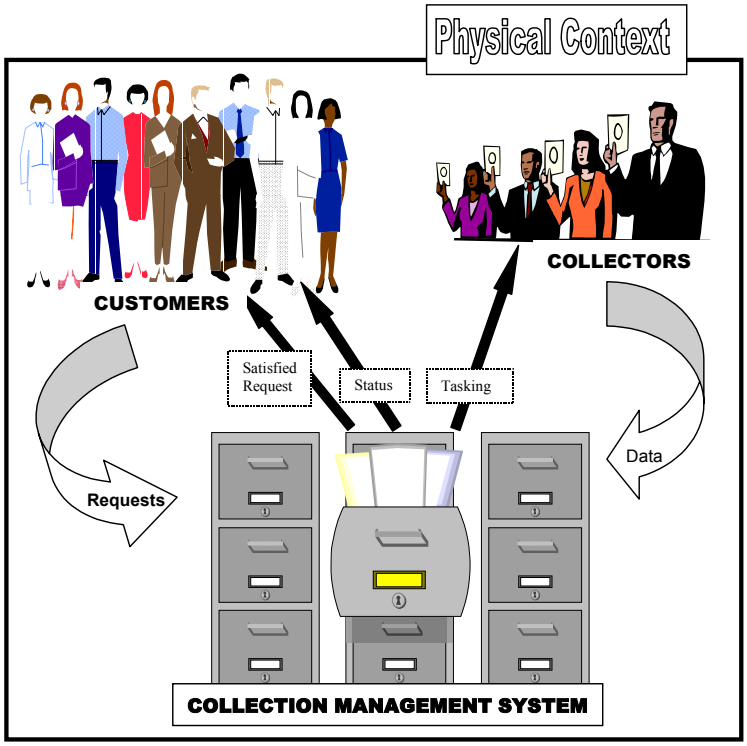
1. We will capture the *source document*, which is the starting point for top-down engineering.
2. We will capture the *originating requirements* from the source documentation.
3. We will *define our system and its boundaries*.
4. We will *derive the system behavior (functional) model* while *extending the physical architecture and allocating all behavior onto the physical architecture*.

By doing this we will establish and maintain traceability between and among the relevant design elements, identify and resolve critical issues, and provide documentation as we progress.

The Sample Problem

In this guide, we examine a sample database for a **Collection Management System**. This system can be thought of as a simple library or resource center. The context diagram below provides a high-level view of the system we will use throughout this guide. You may find it helpful to refer back to this diagram as you build the system structure.

We have a database of collected information.



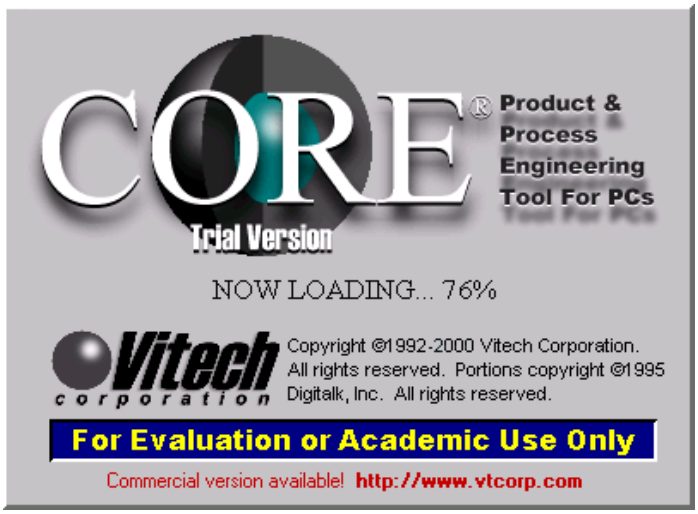
The Collection Management System

The **Collection Management System** controls the inflow and outflow of this collection. **Customers** request information from the collection. The **Collection Management System** processes the customer requests and either returns information to them or provides a status of when they can expect a response. When the requested information is not in the system, the **Collection Management System** tasks **Collectors** to locate the information. (We can think of *Collectors* as Reporters or Researchers.) When the *Collectors* locate information, they pass the data to the *Collection Management System*, which in turn passes it on to satisfy the **Customer** request.

Getting Started with CORE - Opening CORE Trial

Once you have installed the **CORE Trial**, launch the **CORE 3.0 Trial** application.

- Click the **START** button.
- Select the **Programs Menu**, proceed to the **CORE 3.0 Trial** menu, and click **CORE 3.0 Trial**.

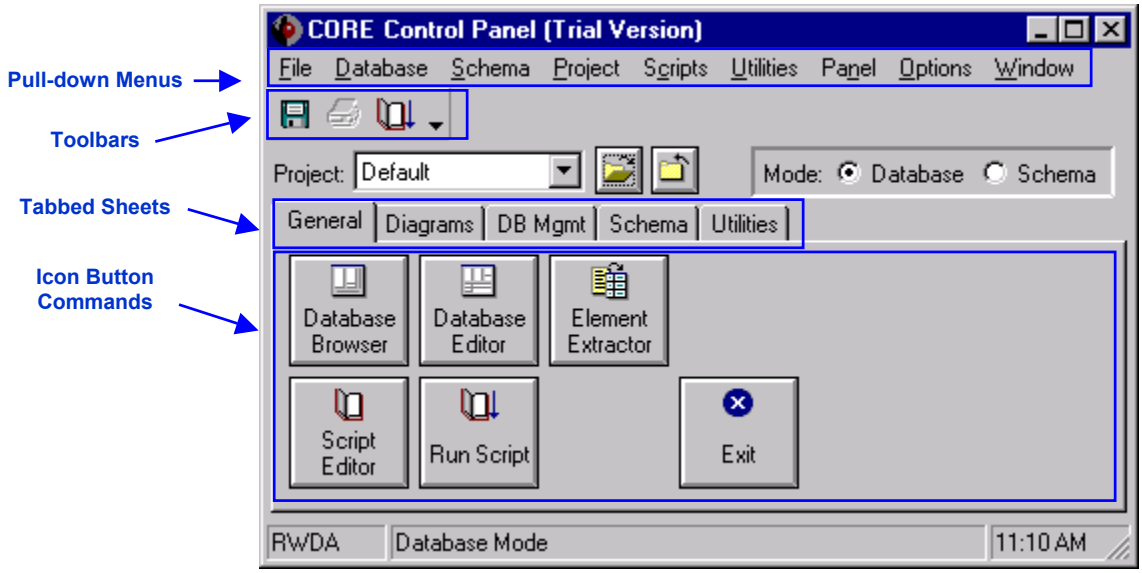


CORE Control Panel

CORE opens with the **CORE Control Panel**. The **CORE Control Panel** provides *pull-down menu* and *toolbar* access to key user interface windows and access to the main menu commands. It is from this window that you navigate your **CORE** database.

New in **CORE 3.0** is the addition of *tabbed sheets*, *toolbars*, and *icons*. The **CORE Control Panel** contains *tabbed sheets* grouping common user commands. Click on a *tab* to access the *icon button commands* for that *tab*. *Toolbars* are present in all **CORE** windows and contain *icons* to access frequently used commands.

CORE Control Panel Window

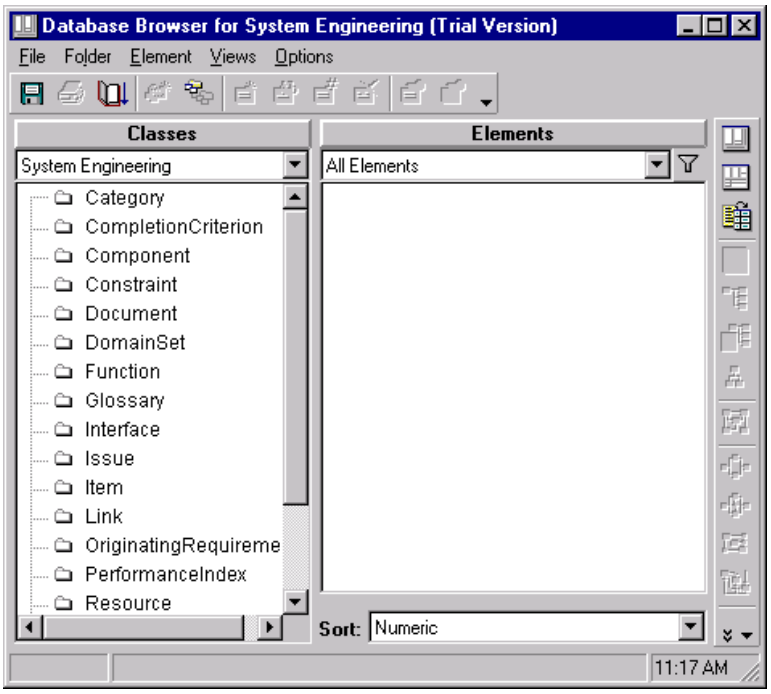


Database Browser

In general, the **DATABASE BROWSER** command opens a window to view the data structure, the **DATABASE EDITOR** command opens a window to view the data structure and to make changes, and the **ELEMENT EXTRACTOR** command opens a window allowing you to extract data from an electronic document text file into the database.

If you click the **DATABASE BROWSER** icon at this point, you see the **CORE** data structure, but there is no data in the database.

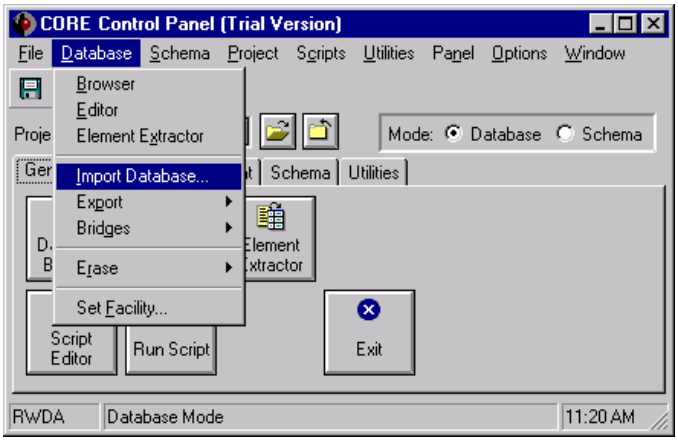
Database Browser – Empty



- Close the **Database Browser Window**

Importing CORE Data

To introduce you to **CORE**, we begin by importing some data. We will use a data file that was created in **CORE** and then exported to the Samples directory of our file system. In general, this import/export capability of **CORE** allows you to transfer a **CORE** database from one computer to another or to make a backup copy of the data. Here, we want to import data so we can see the **CORE** application with data.



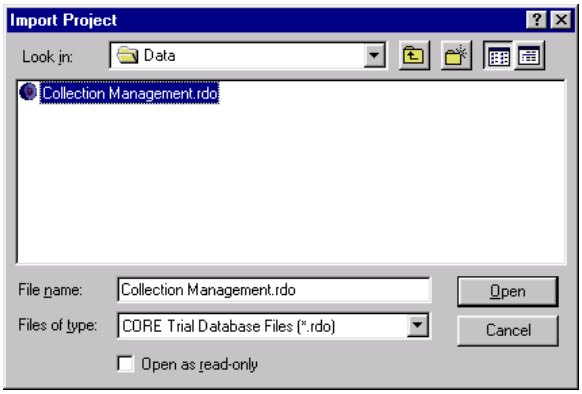
To Import a **CORE** Data File:

- From the **CORE Control Panel**, select **Database > Import Database**.
  (or you can click the **DB MGMT** tabbed sheet and click **Import Database** button.)

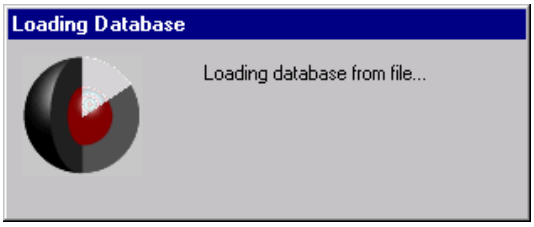
This opens the *Import Database File* dialog.

Note
By default, the Import Database File dialog directs you to the Data directory since this is typically where data files are stored. However, in the Trial, we have data in a Samples directory.

- Navigate to the **CORE30 Trial/Samples** directory and highlight the file named **Collection Management.rdo**.
- Click **OPEN**



The *Loading Database* dialog indicates activity while importing a file.



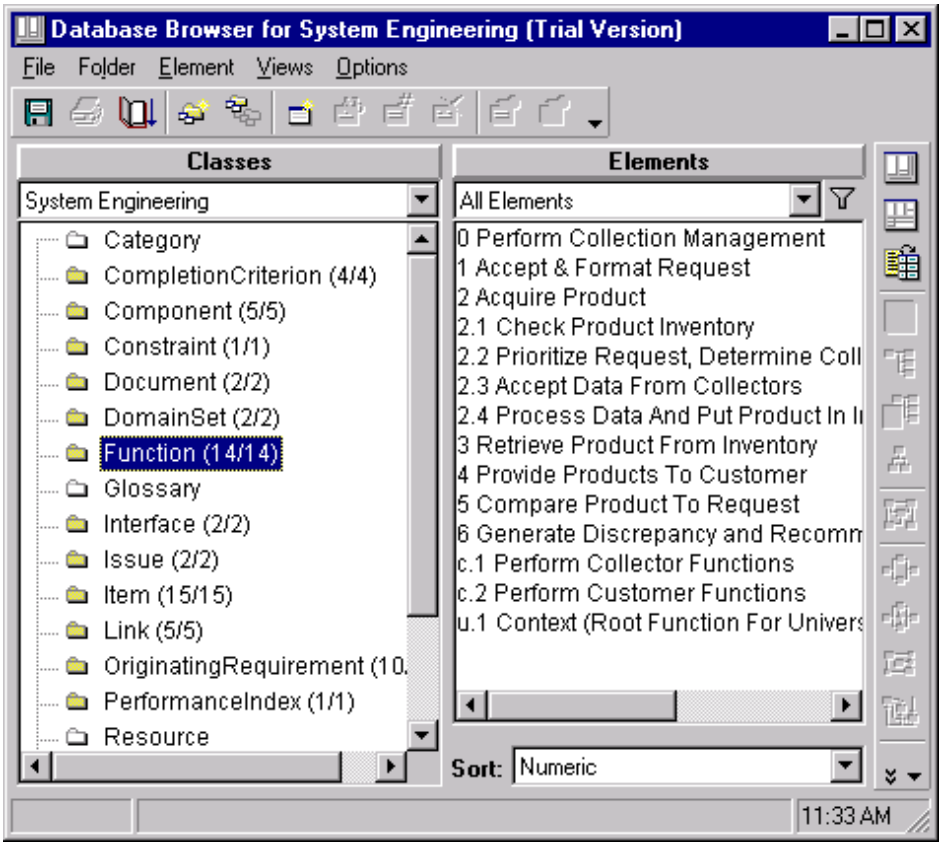
Opening the Database Browser

Now that we have imported the file *Collection Management.rdo*, let us look at what we have in the database.

- From the **CORE Control Panel**, select **Database > Browser** to open a *Database Browser* window.

The left pane lists the *classes* that are used to describe the system.

Database Browser – Loaded



Notice the data that was added when we imported the *Collection Management.rdo* file. We have selected the *Function* class to list the elements in the Function class. Double-clicking on an element name will instantly open a *Text View* window of that element.

- From the **Database Browser**, select **Function** from the **Classes** pane.

Note
You can adjust the size of the Classes pane and Elements pane. A yellow folder preceding the class name indicates that at least one element (instance) of that class has been defined. The numbers in parenthesis indicate how many elements have been defined for that class and how many total elements in all subcategories of the class).

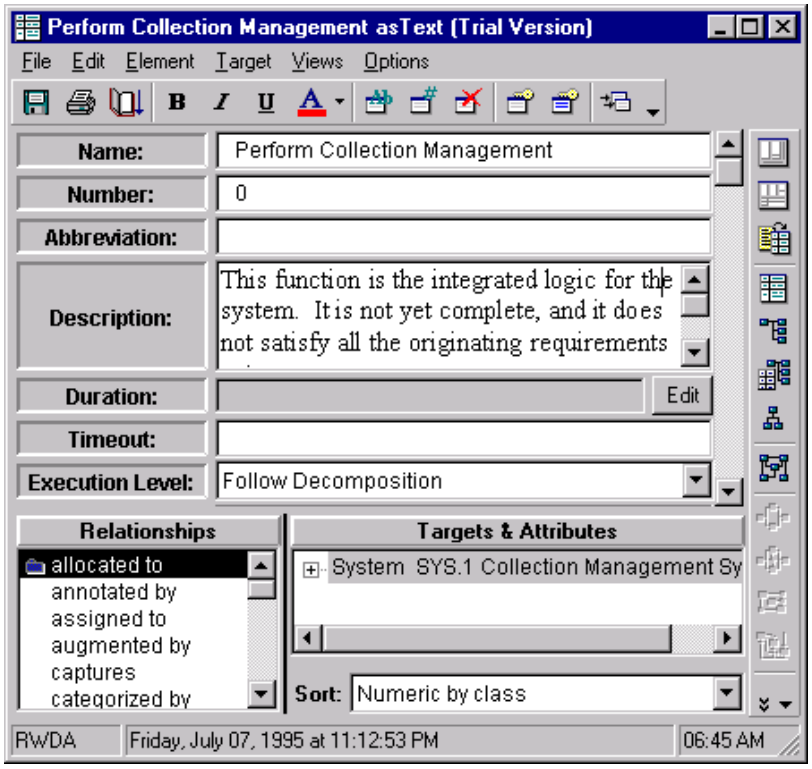
Opening a Text View

A **Text** *View* window provides the complete definition of a given element in the database by displaying all the *attribute values* and *relationship settings*. You can open a **Text** *View* of any database element to *view*, *add*, or *make changes* to the *attributes* and *relationships* of the displayed element. The attributes and their values are displayed in the upper portion of the window. The *relationships* and *targets* that complete the element definition are displayed in the lower portion of the window. Use the *scrollbars* on the right to view the complete list of *attributes* and *relationships*, respectively.

The list of *attributes* and *relationships* differs depending on the *class* of the element displayed. Here we are looking at an *Element* of the class named *Function*, so these attributes and relationships pertain to *Functions*.

- Double-click **Perform Collection Management** from the list of Function elements to open a **Text** *View* window of this element.

Text View of Perform Collection Management



Scroll through the Attributes

Scroll through the list of possible *Relationships*

- A ⊞ indicates a *target attribute*. Click the ⊞ to expand the list and view the attributes of the target.

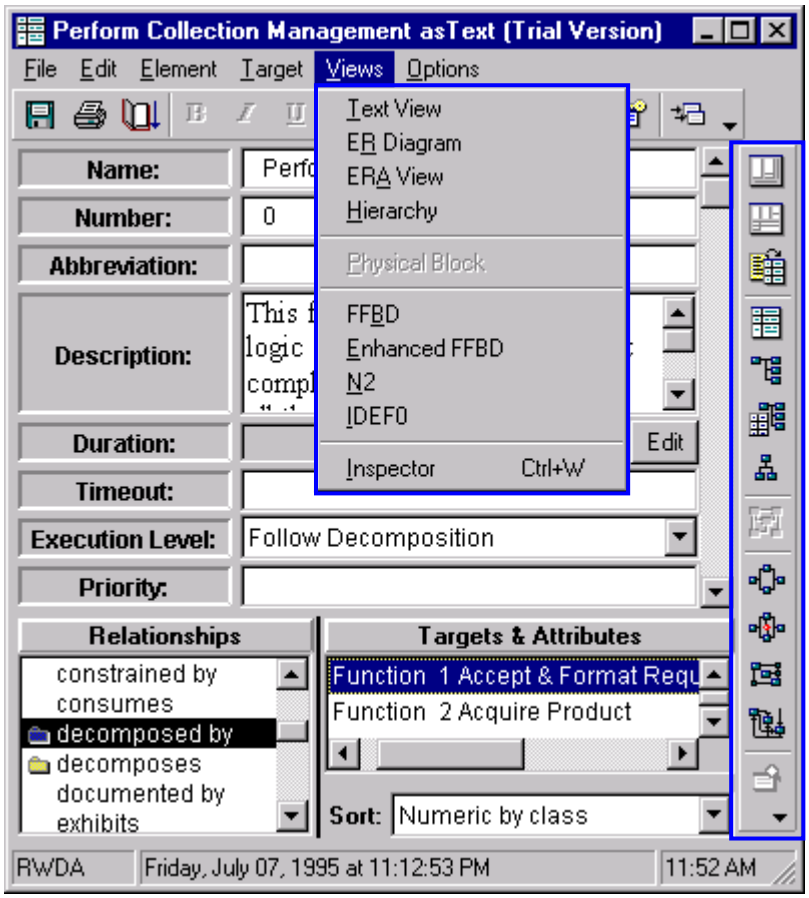
Note
You can adjust the size of the *Relationships* and *Targets & Attributes* panes.

Viewing Class Elements

We viewed the *Perform Collection Management* element attributes by double-clicking the element name in the **Database Browser** window and opening a **Text View** of the selected element.

For other views of the data, select the desired view from the **Views** menu or use the **Database Views** toolbar icons. The views listed in the **Views** pull-down menu are the same as those displayed in the **Database Views** toolbar.

Views Pull-down Menu



Database Views Toolbar

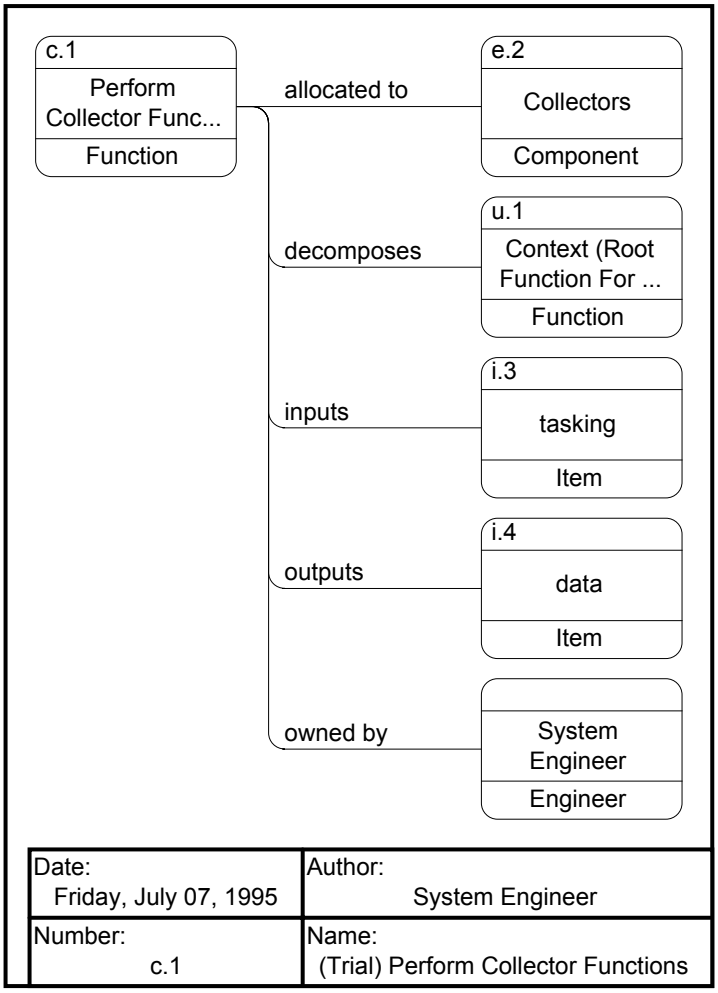
Viewing ER and ERA Diagrams

ER View

- Select **Perform Collector Functions** from the list of elements in the *Function* class
- Click **ER**  to open an **ER View**

The **ER View** is a graphical representation of a selected element. The element and its targets are represented as icons linked by relations.

Element Relationship (ER) View



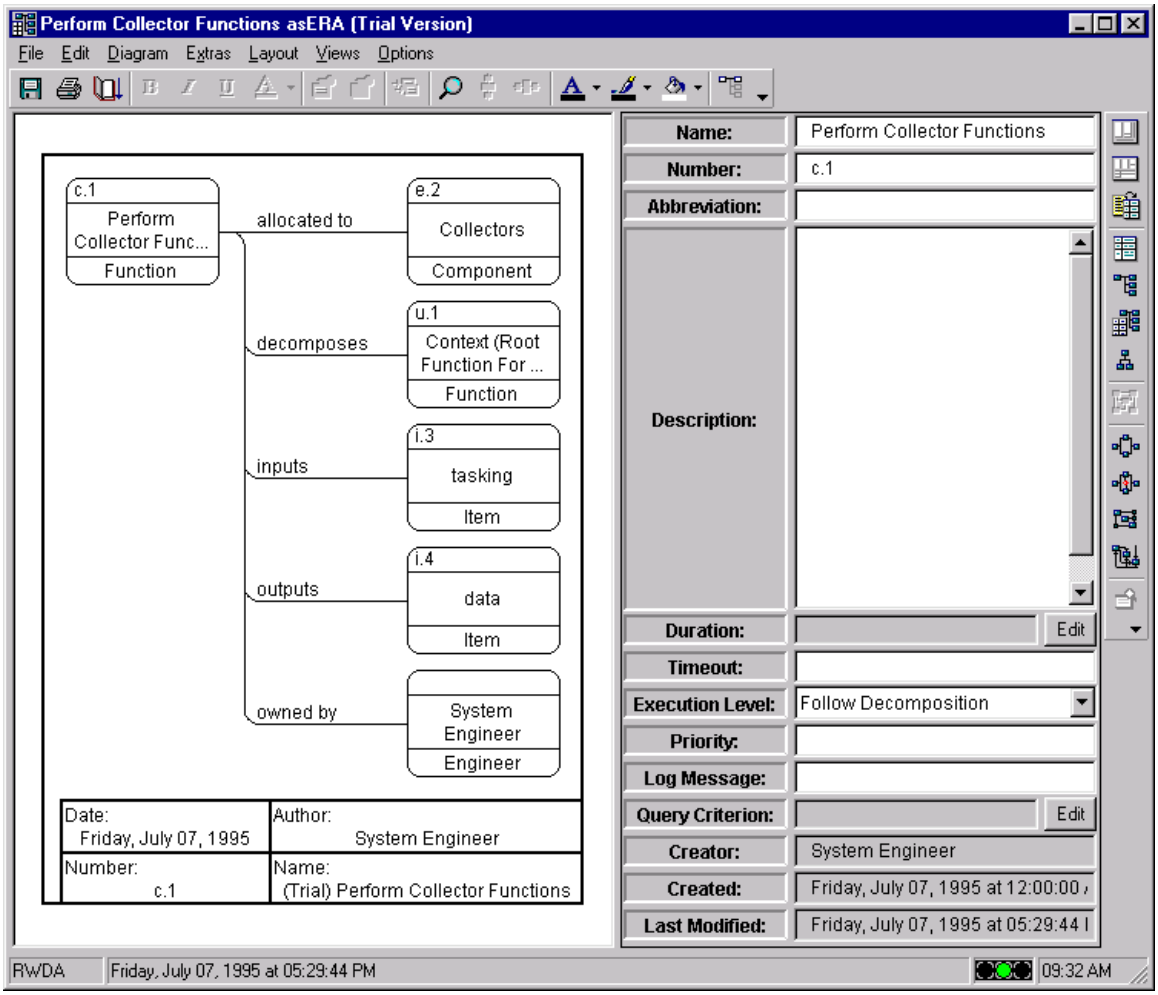
- Close **ER View**

ERA View

The **ERA View** provides a composite of the **ER** and **Text Views**. The **ERA View** completely specifies an element by incorporating the strength of the graphical relation representation with the value of the tabular attribute representation. By having the views together, the user can manipulate *relationships* and *attributes* for the selected element.

- Click **ERA**  to open an **ERA View**. You may find you need to resize or scroll through the window to see all the data.
- Explore more of the database by selecting other *Classes* and *Elements* and opening their respective views.

Element Relationship Attribute (ERA) View

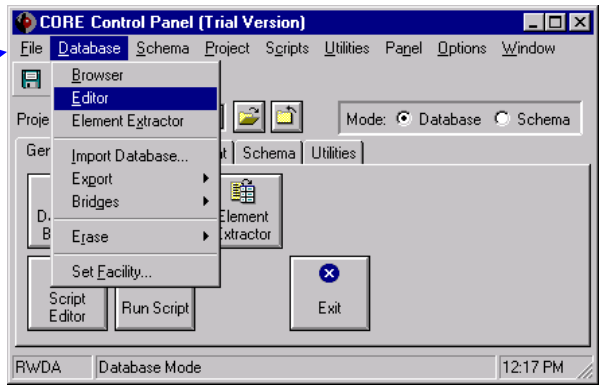


- Close **ERA View**

Opening the Database Editor

Another way to view and make changes to the data in the **CORE** database is with the **Database Editor.** The **Database Editor** combines the **Browser** and **Text View** windows into one— allowing you to view the structure of the data and make updates to the selected element in the same window.

- From the **CORE Control Panel**, click **DATABASE EDITOR** to open a *Database Editor* window.
- Take a moment or two to familiarize yourself with the layout of the Database Editor window.



The scrollable left pane of the **Database Editor** lists the classes that are used to describe the system.

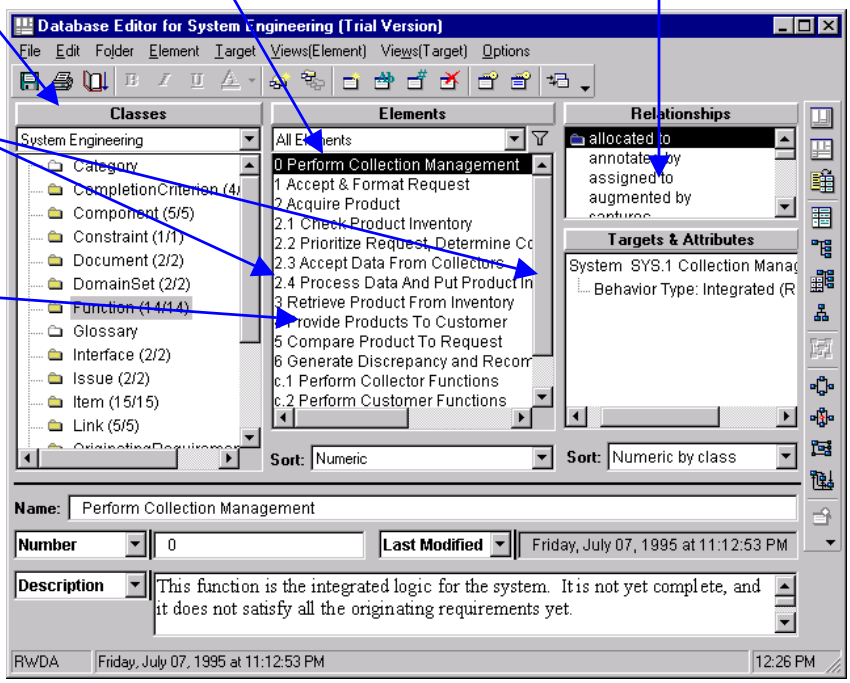
The scrollable middle pane lists the corresponding elements of the selected class.

The scrollable right pane lists the possible relationships that can be set for the selected class.

You can adjust the size of the panes.

Instead of opening a **Text View** on an element, you can select the element from the **Database Editor** and view/update the attributes in the same window.

Use the pull-down menus to access other attributes from the complete list of attributes.



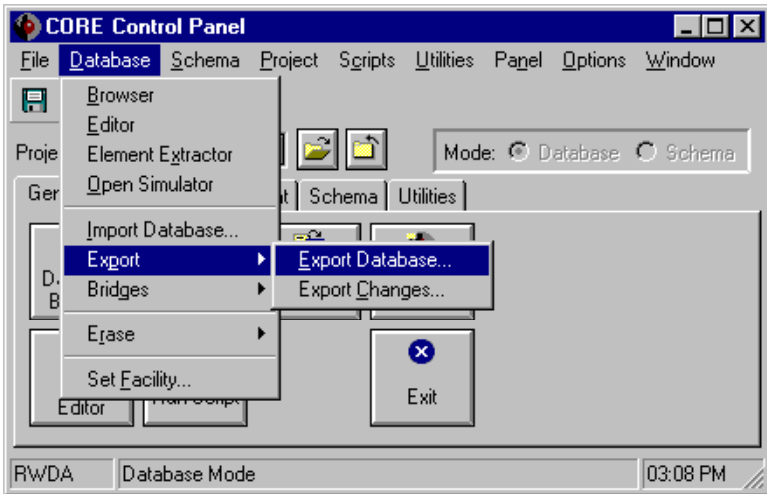
Database Editor

Saving CORE Data

In the **CORE Trial Version**, we import and export the database with an *.RDO* extension.

Note

The actual CORE version imports and exports the CORE database with an .RDT file extension. The full version of CORE can also save an image with a .COR file extension allowing it to load faster. This feature is not accessible in the Trial version.



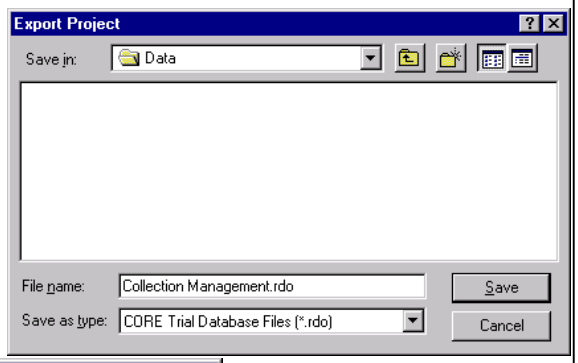
To Save Your *CORE Trial Data* to an *.RDO* File:

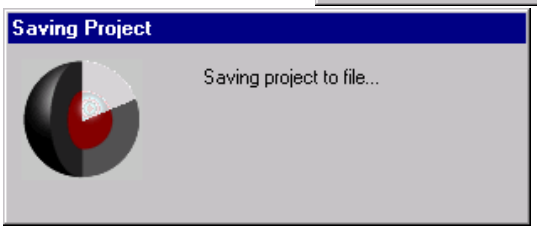
- Select **Database > Export > Export Database** from the **CORE Control Panel** menus to export **CORE** data.

An *Export Database* dialog prompts you for an *RDO* file name in the *Data* directory.

- We will name the export file **Collection Management.rdo**
- Click **SAVE**.

The *Saving Project* dialog indicates activity while exporting a file.





Note

Refer to this section to save/export and load/import your data as you proceed through the rest of the guide. When you need a break, or need to exit the CORE Trial, you will want to export you data to an RDO file and then import that RDO data file when you return to CORE in order to continue where you left.

This Page Intentionally Left Blank.

Building a CORE Database

In this section, you build a CORE database from scratch

- **Extract a System Document into CORE**
- **Extract Originating Requirements into CORE**
- **Define Relationships**
- **View the Requirement Hierarchy**

Capturing the Problem and the Originating Requirements

Now that you have seen how to import a *database file* and *view data*, we will see how to *build a database from scratch*.

Before proceeding with this section, be sure to START WITH AN EMPTY DATABASE. (**CORE Trial** is empty each time you launch the application.)

- Close any open **CORE** windows (except the **CORE Control Panel**).
- From the **CORE Control Panel** menus, select **Database > Erase > Erase Database**.
- Answer **Yes** to the warning.

We will start by capturing the *source document*. We want to put our entire source document into **CORE**. The **Element Extractor** is an easy way to transfer text from a text-based file into the **CORE** database structure. New with **CORE 3.0**, the **Element Extractor** now accommodates formatted text in either *RTF*, *HTML*, or *TXT* formats.

To accomplish this, we use **CORE's Element Extractor**.

- From the **CORE Control Panel > General** tabbed sheet, click the **ELEMENT EXTRACTOR** button.

For extracting, you need to load your source document. The source document must be DOC, RTF, HTML, HTM, or TXT file types.

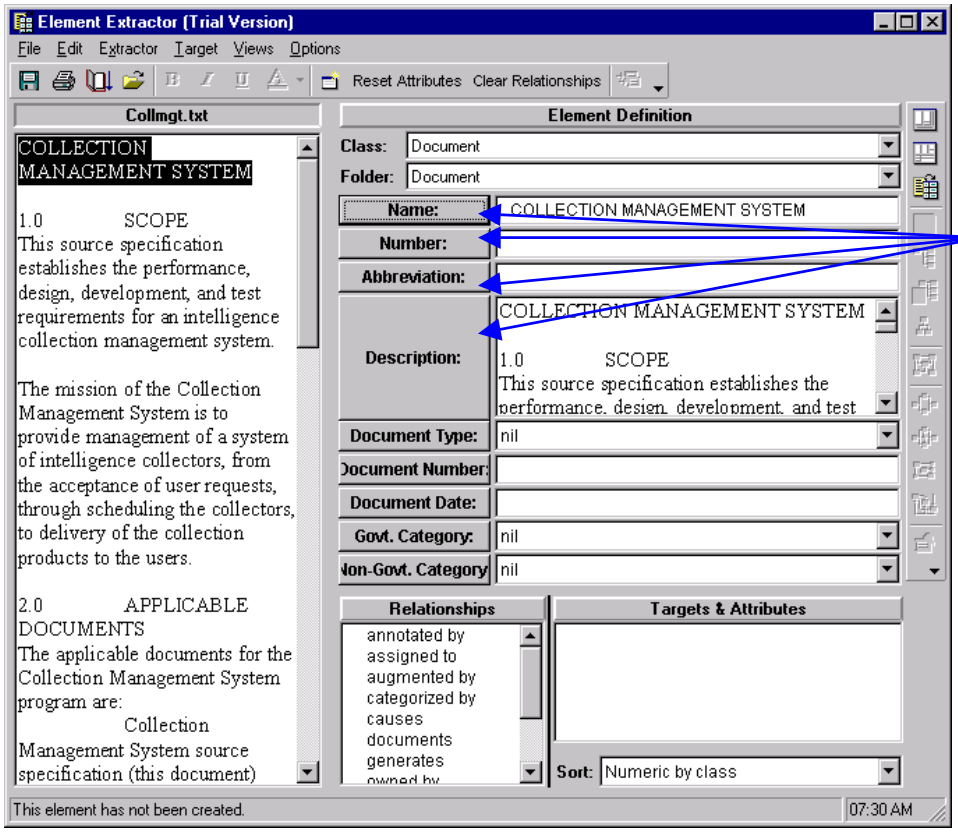
- Select **File** > **Load Document**.

We will use the file *Collmgt.txt*, located in the *Samples* folder of the **CORE Trial** directory.

- Navigate to the **Samples** subdirectory
- Change the default *Files of Type* to **Text Files (*.txt)**
- Highlight **Collmgt.txt**.
- Click **OPEN**.

The *Collmgt.txt* file displays in the left pane while the element being extracted/created is built and defined in the right pane. The *attributes* and *relationship* fields vary depending on which *class* is selected from the *class pull-down selection list*.

Element Extractor Window



The **Element Extractor** can extract text into any class of elements.

- Select the **Document** class from the **Class** pull-down selection list.

Many of the attributes for an Element can come directly from the source document. In general, to move text from the left pane to the element attribute fields, highlight the desired text in the left pane and press the transfer button corresponding to the desired attribute.

To add a *Description* to the Document element:

- Resize the **Element Extractor** window in preparation for reviewing the source material and loading it into the system design database.
- Highlight the entire contents of the *Collmgt.txt* file (**Ctrl+A**)
- Click the **DESCRIPTION** *transfer button* in the right pane to transfer the selected text to the *Description* attribute field.

To add a *Name* to the Document element:

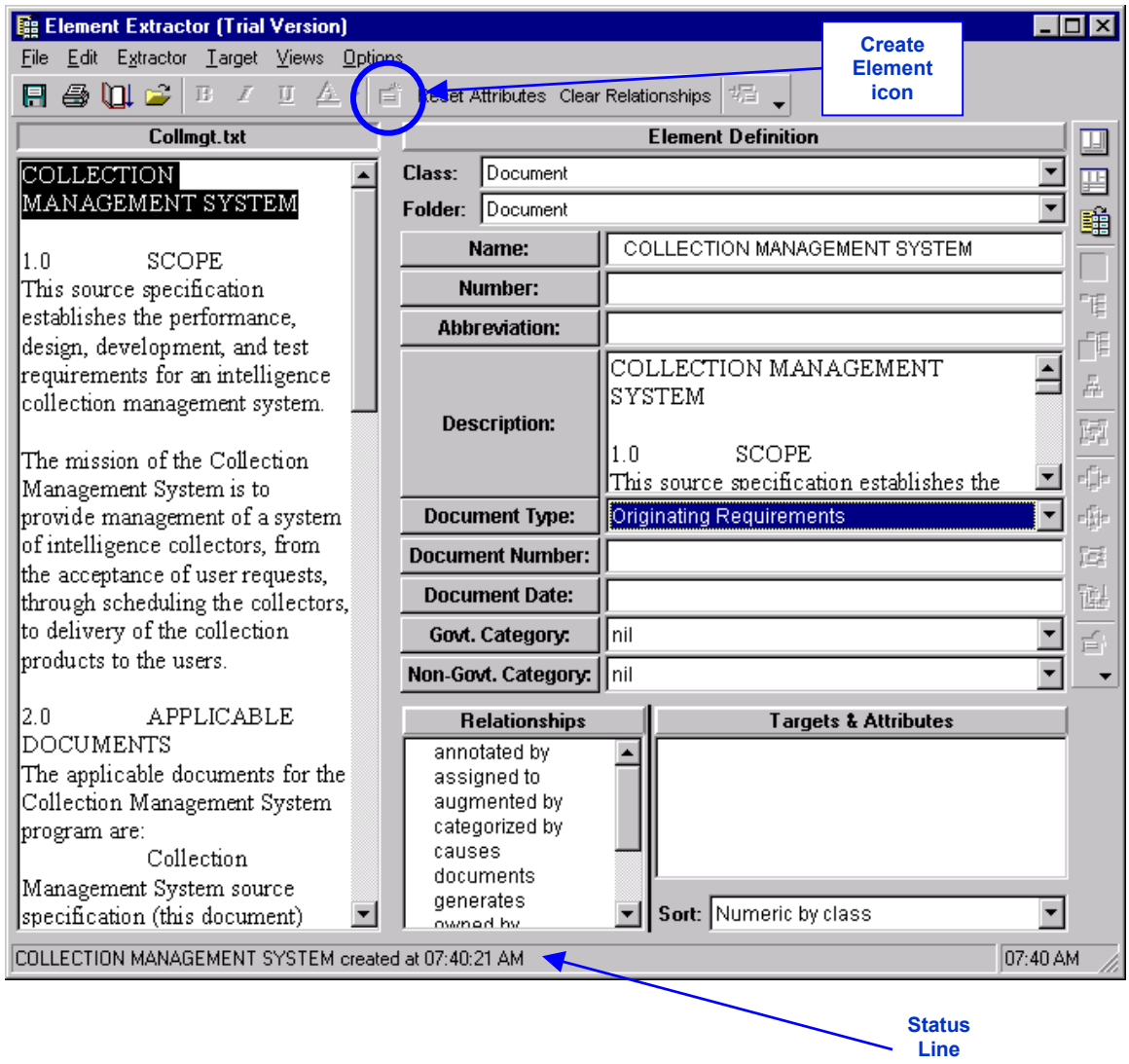
- In the Name field, type **COLLECTION MANAGEMENT SYSTEM,** or highlight the text *COLLECTION MANAGEMENT SYSTEM* in the left pane and click the **NAME** *transfer button* in the right pane to transfer the text to that attribute field.

Saving the Document Element in the Design Repository

The *Document Type* attribute is set by choosing from a predefined list of possible values (an enumerated list).

- Click on the *down arrow* next to the *Document Type* field, and choose **Originating Requirements** to reflect the source of this information.

Element Extractor Window with Document Type Set to Originating Requirements



Once all desired attributes are defined, we need to enter the *Document Element* into the design repository (database).

- From the **Element Extractor** window, click the **CREATE ELEMENT** icon on the **Element Extractor** toolbar to enter the *Element* in the database (or select **Extractor > Create Element**). The status line reflects when an element has been saved to the database.

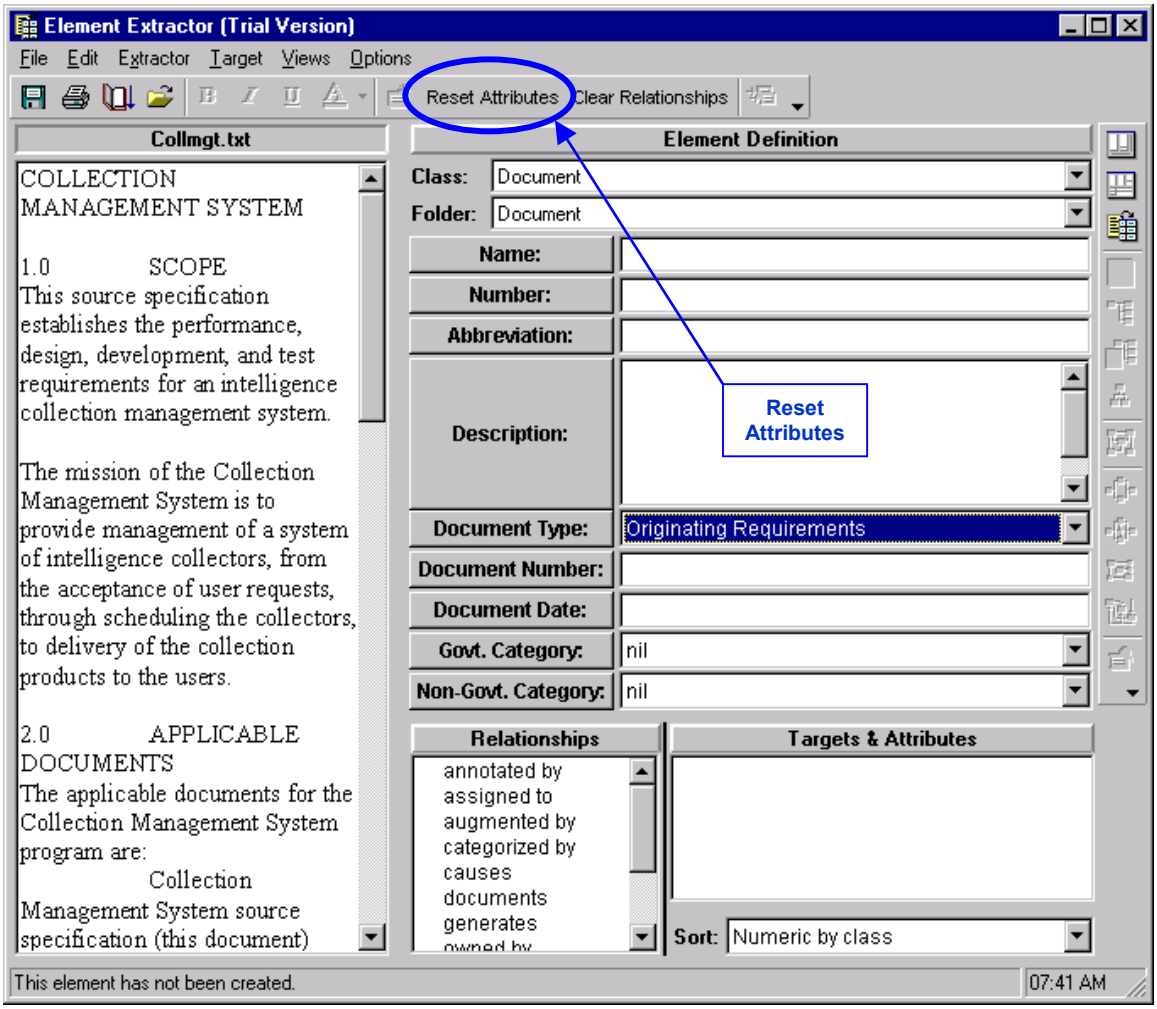
We have finished defining the *attributes* of the *Document* element we named *Collection Management System.* We will talk about defining *relationships* between *elements* in the next few pages when we extract the *Originating Requirements Element*.

Extracting Originating Requirements

Our next step is to extract and define our *Originating Requirements* elements from the source document, *Collmgt.txt*. Notice that after we saved the Document element to the database (selected **Create Element**), the *attribute data* is still displayed in the fields. This saves you from having to re-enter it if you want to make use of some of the same data. In our case, at this point, we won't so we will want to clear the attribute fields before proceeding.

- Click the **Reset Attributes** command on the toolbar (or **Extractor > Reset Attributes** from the menus) to clear previous data from the fields.
- From the *Document Type* selection list select **Originating Requirements**.

Element Extractor Window with Reset Attributes Selected



The *attribute* and *relationship* fields will reflect attributes and relationships for an *Originating Requirement* element.

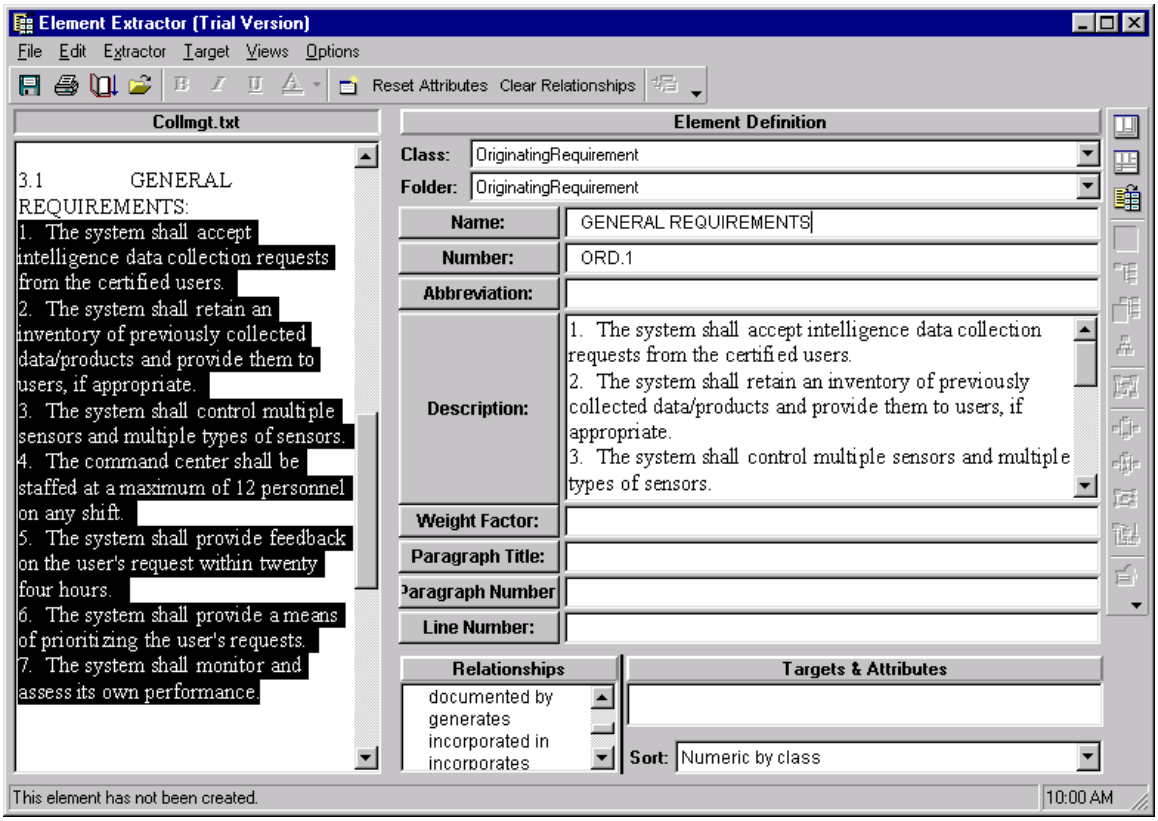
Extracting the Top-Level Requirement

During this extraction process, we want to establish a *hierarchy of requirements*. In our case, the *General Requirements* element incorporates a number of subordinate requirements. We will first establish a parent, or top-level, *Originating Requirement* from the GENERAL REQUIREMENTS section of the *Collmgt.txt* file. You may need to scroll down in the left pane to see this section of the text file.

- Highlight the words **GENERAL REQUIREMENTS** from the text in Collmgt.txt.
- Click the **NAME** *transfer button* to insert the text into the *Name* field, which saves you from having to type the name in the field.
- Type **ORD.1** in the *Number* field. (There was no short way to enter this data.)
- Highlight the text in *Collmgt.txt* within the heading **GENERAL REQUIREMENTS**, as shown in the figure below.
- Click the **DESCRIPTION** *transfer button* to insert the text into the *Description* field.

To be thorough, you can enter data in the other attribute fields as appropriate.

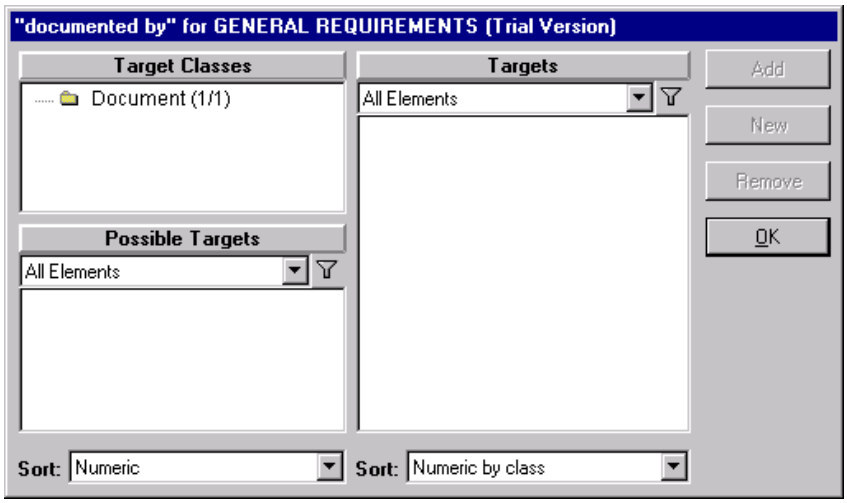
General Requirements – ORD.1



Defining a Relationship

For *traceability*, we want to establish that this **Originating Requirement** element (**ORD.1**) is documented by the Document element named *Collection Management System*. The documented by relationship identifies the source document which specifies and/or enhances the definition of the element.

- Double-click the **documented by** relationship in the Relationship pane (or select **Target > Edit Target** from the menus) to open a *Target Dialog* for the relationship.



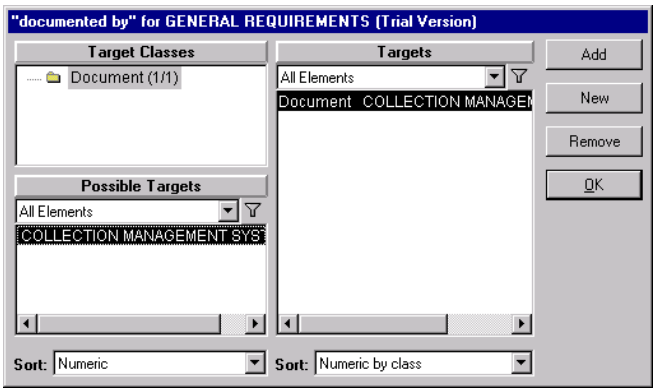
The **Edit Target** command is used to add one or more targets to the selected relationship. It *automatically creates the complimentary relation* linking the target with the selected elements.

Defining a Relationship (cont.)

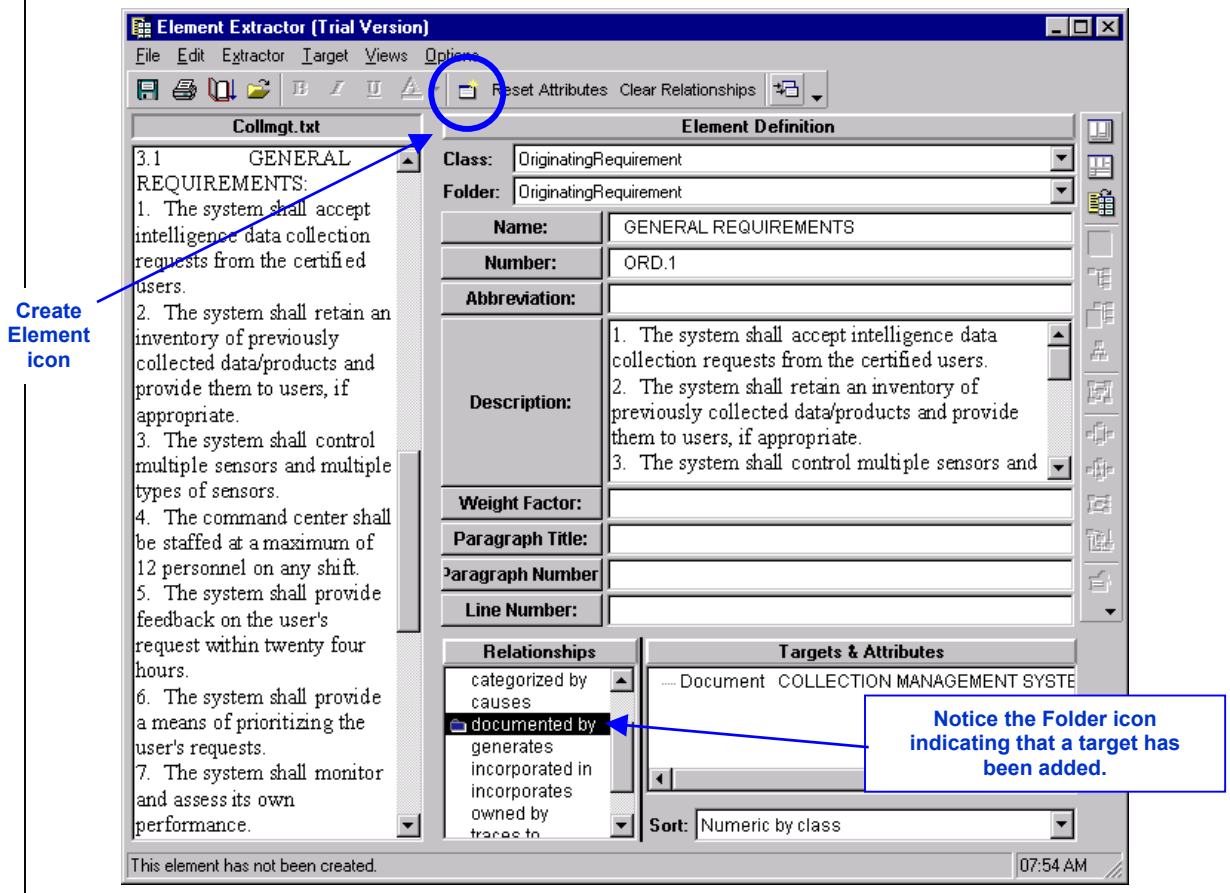
The *Target Dialog* allows you to add or modify *relationships*. It lists the allowable target classes for the specified relation, which in this case is documented by. With a *Target Class* selected, its possible targets are listed. To make a *Possible Target* a *Target*, select a possible target and click **ADD** (or double-click on the possible target). As soon as the target is added, the new target is added to the *target* list. You can also click the **NEW** button to create a new target, if you want one that does not already exist.

Here, we will use the one we have.

- From the **Target Classes**, select **Document**
- From the **Possible Targets**, select **Collection Management System**
- Click the **ADD** button to add the document element to the targets list.
- Click **OK** to close the *Target Dialog*.



Returning to the **Element Extractor**, notice the target added to the documented by relationship. We are now ready to enter this *Originating Requirement* in the repository.



- From the **Extractor** *pull-down menu*, click **CREATE ELEMENT** (or **Ctrl+E**)

Note: You can also click the **Create Element** icon.

Extracting the Child-Level Originating Requirements

Now that we have defined the top-level **Originating Requirement**, we will break out each of the seven General Requirements to create seven individual, child-level *Originating Requirements*. These requirements are incorporated in their parent, which we named **GENERAL REQUIREMENTS**. *Traceability* back to the source document, *Collection Management System*, is achieved through the parent (as defined with the documented by relationship).

To begin, we took the entire *General Requirements* section of our *Collmgt.txt* and placed it in an *Originating Requirement* class element that we numbered ORD.1; this will serve as our parent level. Now we will break up the section, placing each numbered requirement in its own element of the *Originating Requirements* class; this will serve as our child level.

We will give each of the seven requirements a number and name, such as:

ORD.1.1	Accept Requests
ORD.1.2	Retain Inventory
ORD.1.3	Control Multiple Sensors
ORD.1.4	Maximum Staff
ORD.1.5	Provide Feedback
ORD.1.6	Prioritize Requests
ORD.1.7	Monitor and Assess

Since we are creating child-level elements to the same class (*Originating Requirements*), we don't need to select another class from the selection list. All we need to do is reset the *attributes* and *relationships*.

- Click the **Reset Attributes** button to clear all the attribute fields.
- Click the **Clear Relationships** button to clear all the relationship fields.

Now we have a clean slate to use to begin transferring text and thus creating the first of our seven child elements.

Extracting the Child Level Originating Requirements

1 Type a name in the *Name* field. Using the list on the previous page, the name of the first child element is **Accept Requests**.

2 Establish a hierarchical numbering system for the child elements. Number the first element **ORD.1.1**. (The second will be **ORD.1.2** etc. as shown in the list on the previous page.)

3 Highlight the text for the description from the *Collmgt.txt* file. For the first child element, we highlight the text in the first *General Requirement*.

4 Click the **DESCRIPTION** *transfer button* to place the text in the *Description* field.

Establish the incorporated in relationship to its parent, **GENERAL REQUIREMENTS**.

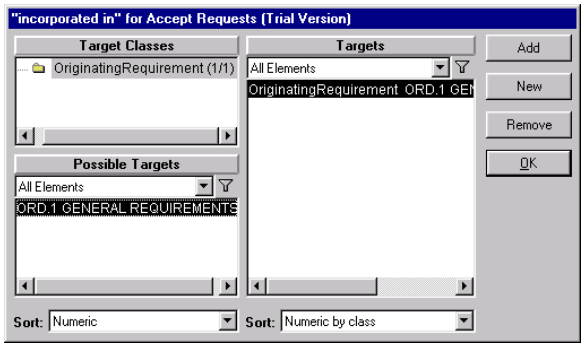
5 Double-click **incorporated in** from the *Relationships* pane to open the *Target* Dialog.

6 Highlight **OriginatingRequirement** from the list of *Target Classes*.

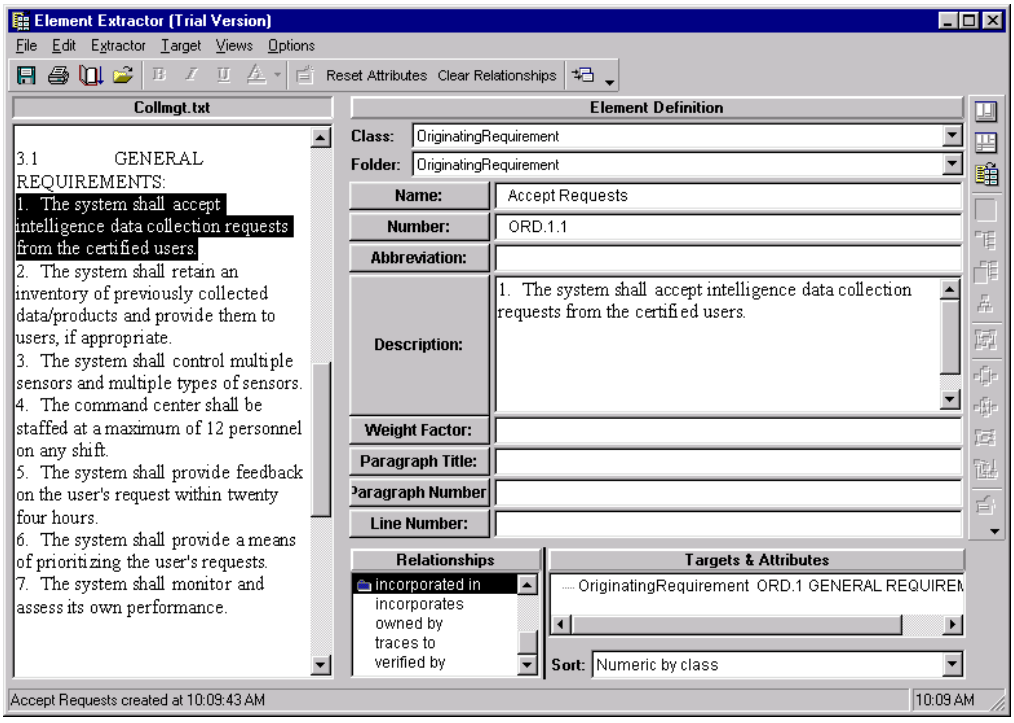
7 Double-click **ORD.1 GENERAL REQUIREMENTS** from the *Possible Targets* to add it as a target.

8 Click **OK** to close the *Target Dialog*.

9 From the **Element Extractor**, click the **CREATE ELEMENT** *icon* to store this element definition.



- Click the **Reset Attributes** to clear the attributes values, but **DO NOT** clear the relationships. The relationship you established for the first child element can be used for the subsequent elements.



Create the other six child elements using the *Numbers* and *Names* from page 28. You can skip steps 5-8 each time through since you didn't clear the relationships.

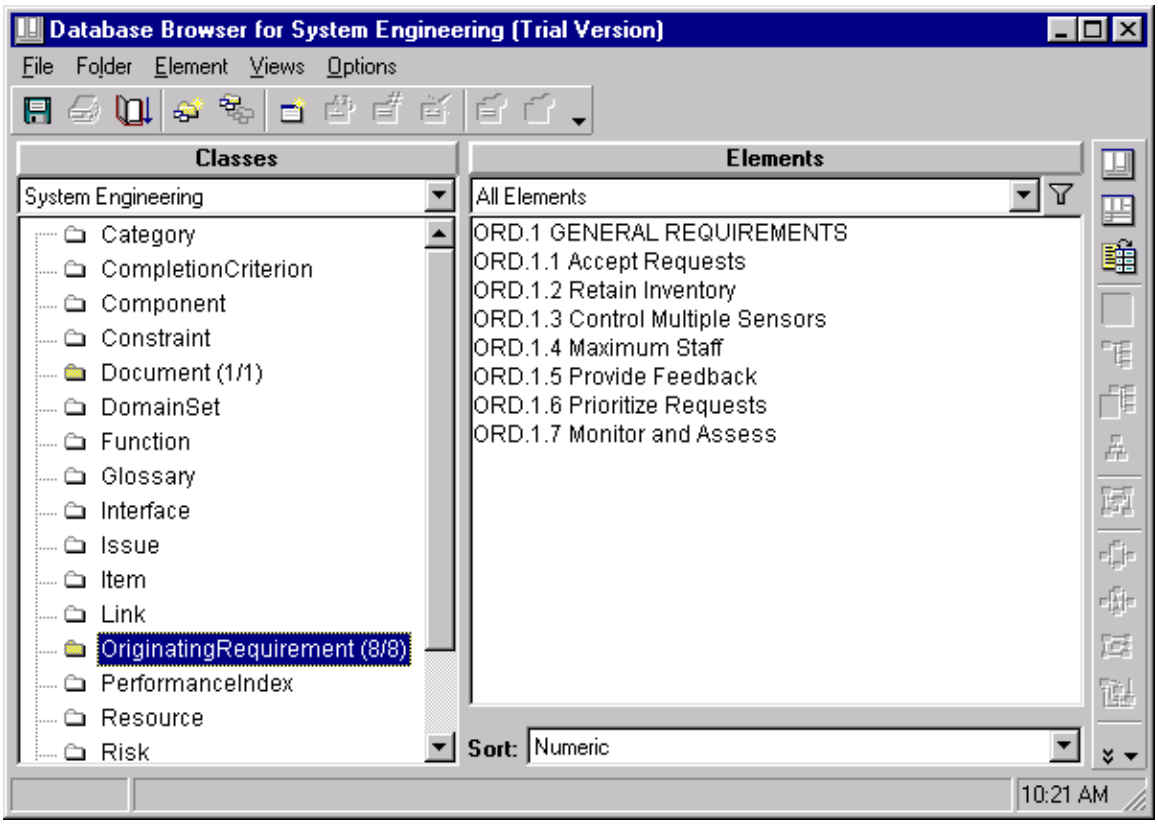
Viewing the Hierarchy in the Browser

Let's check what we have done by looking at the elements we created from a Database Browser window.

We can close the **Element Extractor** window; we are done with it for now.

- Using the **Views** *pull-down menu*, open a **Database Browser** window.
- From a **Database Browser**, select the **Originating Requirements** class.

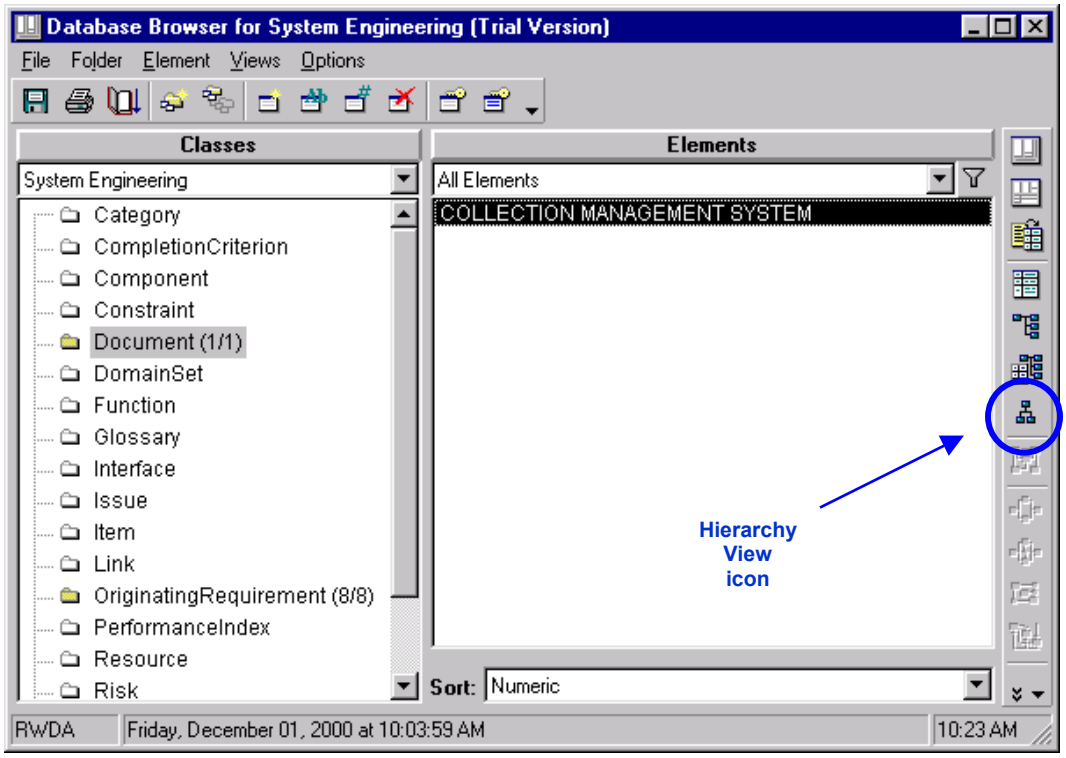
Notice the Elements we created.



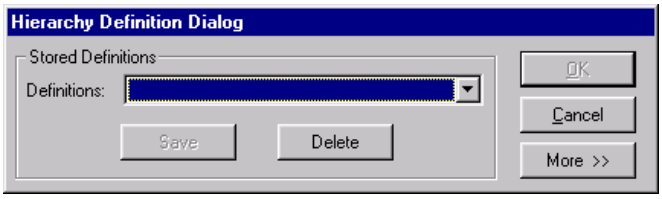
Viewing a Traceability Hierarchy

Now that the seven child-level *Originating Requirement* elements have been created, let us view a *Traceability Hierarchy Diagram* from our source document. Recall that our **Originating Requirements ORD.1** is documented by our source Document named Collection Management System. This means that our Document documents our Originating Requirements. Let's see a diagram of this.

- From the **Database Browser**, select **Document** from the *Classes* pane.
- Select **COLLECTION MANAGEMENT SYSTEM** from the *Elements* pane.

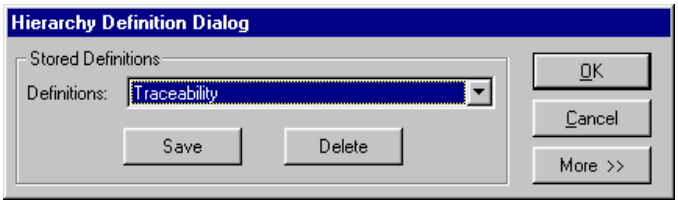


- Click the **HIERARCHY** icon from the **Database Views** *toolbar* or select **Views > Hierarchy** from the menus to open a **Hierarchy** dialog.



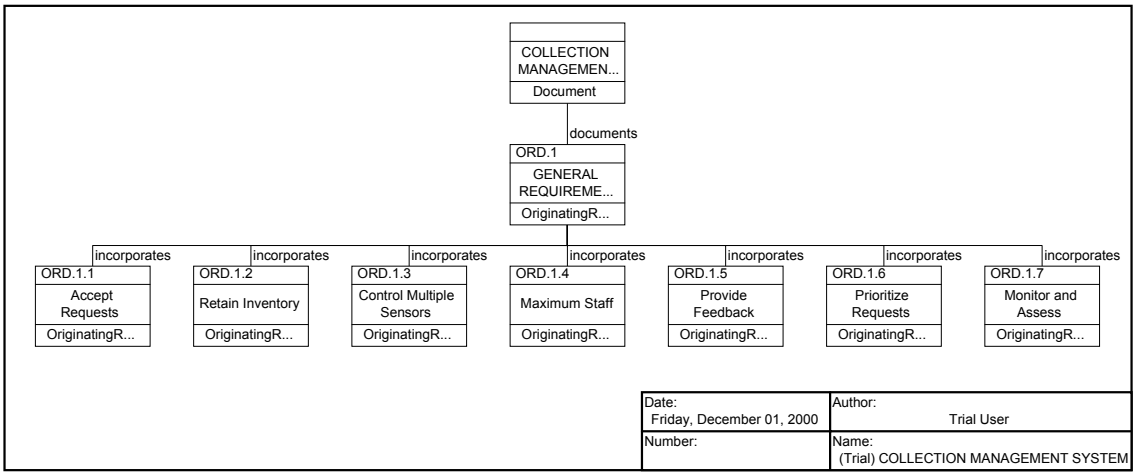
The *Hierarchy Definition Dialog* provides selections for building various hierarchy diagrams. We will select a *Definition* from the enumerated list of stored definitions.

- Click on the *drop down arrow* to show the list of stored definitions and select **Traceability**
- Click **OK**.



Viewing a Traceability Hierarchy Diagram

Your diagram should look similar to the one shown below. Since the **Collection Management System** document was the source from which we extracted the specific requirements, we really have traceability from the source *Document* through the third tier (child-level) *Originating Requirements*.

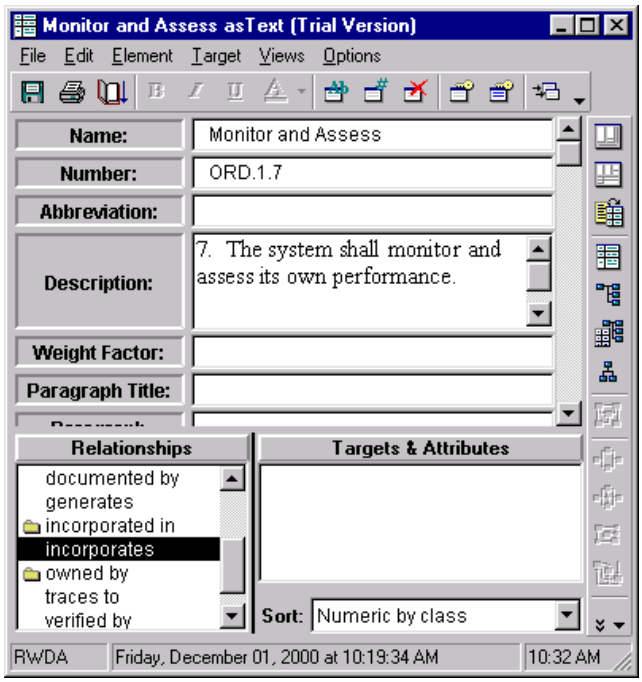


Notice that **ORD.1.7 Monitor and Assess** Originating Requirement can be broken down further since it combines *monitor and assess*. So let's add another level of Originating Requirements by adding *targets* for the *incorporates* relationship of **ORD.1.7 Monitor and Assess**. We will add these elements directly from the **Hierarchy** diagram.

Adding Elements in a Traceability Hierarchy

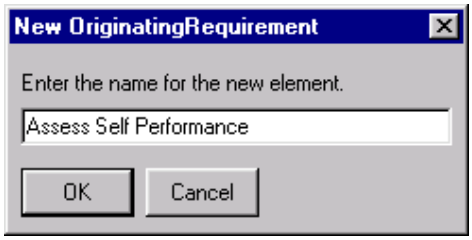
CORE allows us to create elements from a diagram. Using the **Traceability Hierarchy** diagram, we will create two additional originating requirements incorporated in **ORD.1.7** to break down the requirement to its lowest possible level.

- From the **Traceability Hierarchy** diagram, double-click the **ORD.1.7 Monitor and Assess** object to open a *Text View* of this element.
- In the **Text View** window double-click the incorporates relationship to open the **Target Dialog**
- In the **Target Dialog**, select **Originating Requirement** from the list of *Target Classes*.

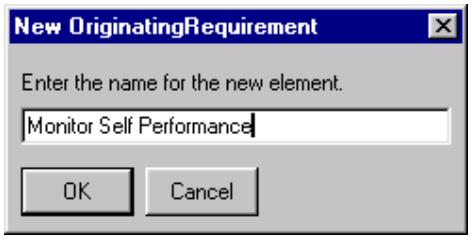


Instead of selecting a target from the list of *Possible Targets*, we will create two new ones.

- Click **NEW**.
- Type a name for the target. We will name one **Assess Self Performance.** Click **OK** and the new target is listed in the *Targets* pane.
- From the **Target Dialog**, click the **NEW** button again to create another target.



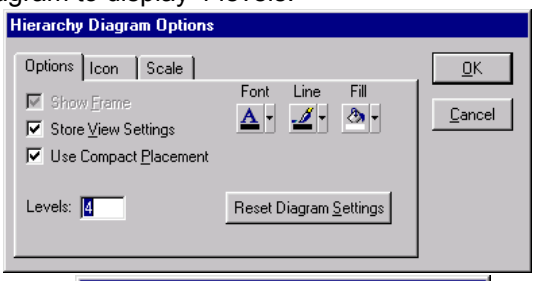
- We will name this one **Monitor Self Performance.** Click **OK** and the new target is listed in the *Targets* pane.
- Click **OK** to close the **Target Dialog**.



Viewing a Traceability Hierarchy

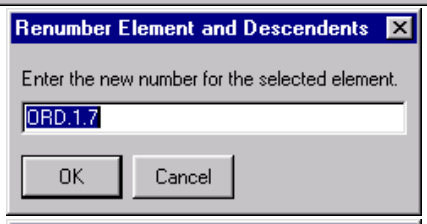
By default, hierarchy diagrams display 3 levels. In order to see these added elements in our **Traceability Hierarchy** diagram we need to change the diagram to display 4 levels.

- From the **Hierarchy** window, select **Options > Local Diagram Options**.
- Type **4** in the Levels field of the **Options** tabbed view of the **Hierarchy Diagram Options** dialog.

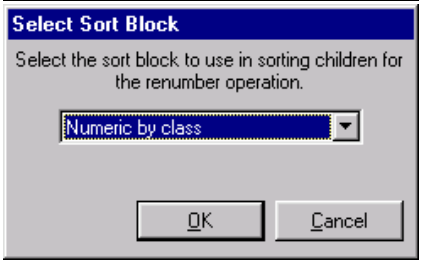


Notice these new elements are not numbered.

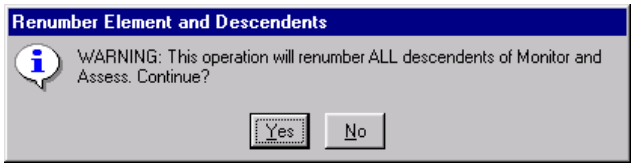
- Right-click on the Monitor and Access element and select **Renumber Selection**.
- **ORD.1.7** appears by default; click **OK**.



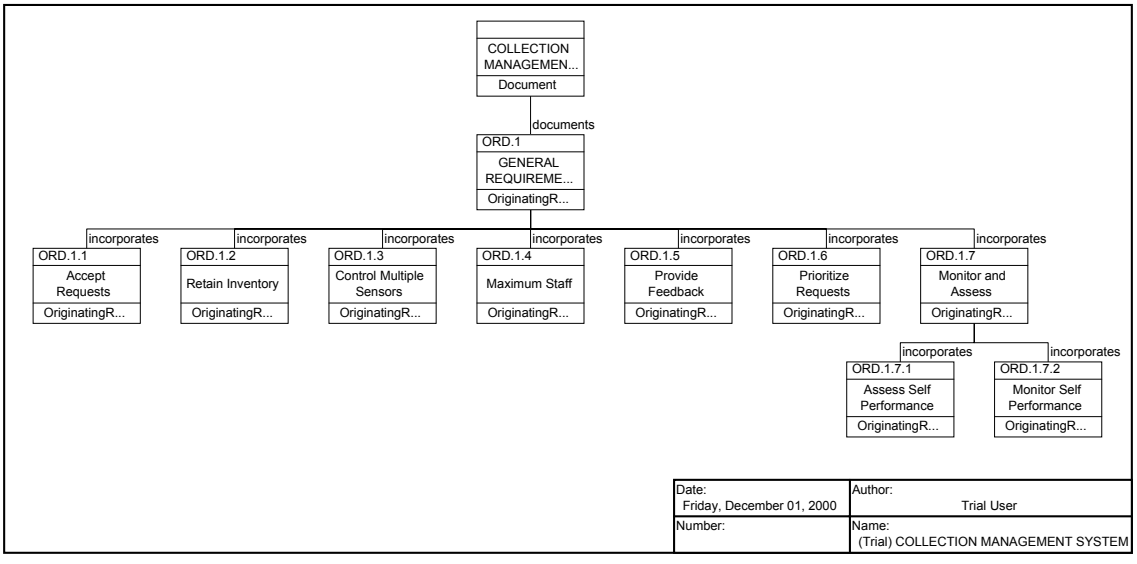
- Click **OK** when prompted for a sort block type.



- Click **Yes** when the warning appears.



Your diagram should look similar to the one shown below.

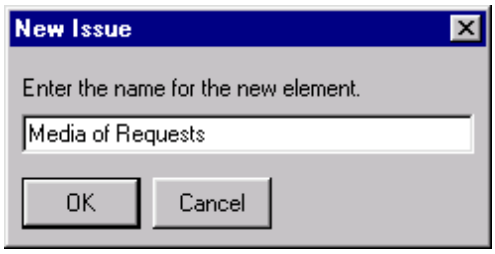


Enhancing the System Definition with Issues

Now that we have extracted the *Originating Requirements*, the requirement analysis begins. As a System Engineer, you want to identify problems encountered during system engineering such as poorly stated or conflicting requirements. In **CORE**, these problems can be captured as *Issues*. An *Issue* identifies a problem (as well as a resolution) with an element in the system design or specification. The primary application is documenting problems with requirements.

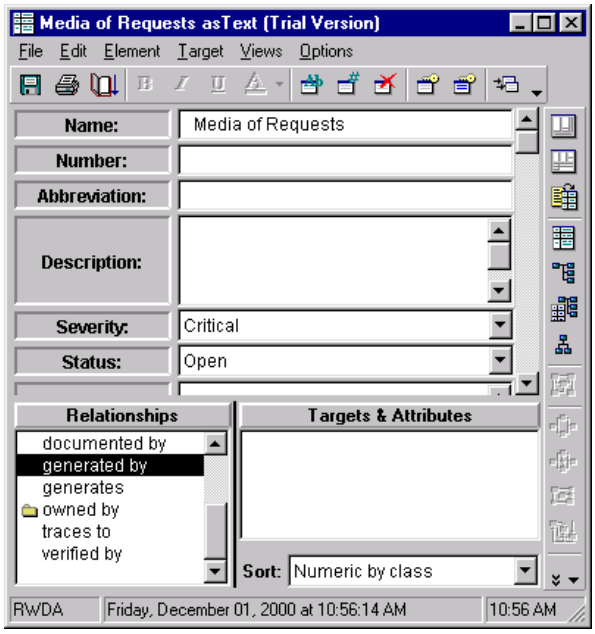
Let's add an *Issue* to ask for clarification on our **Originating Requirement**.

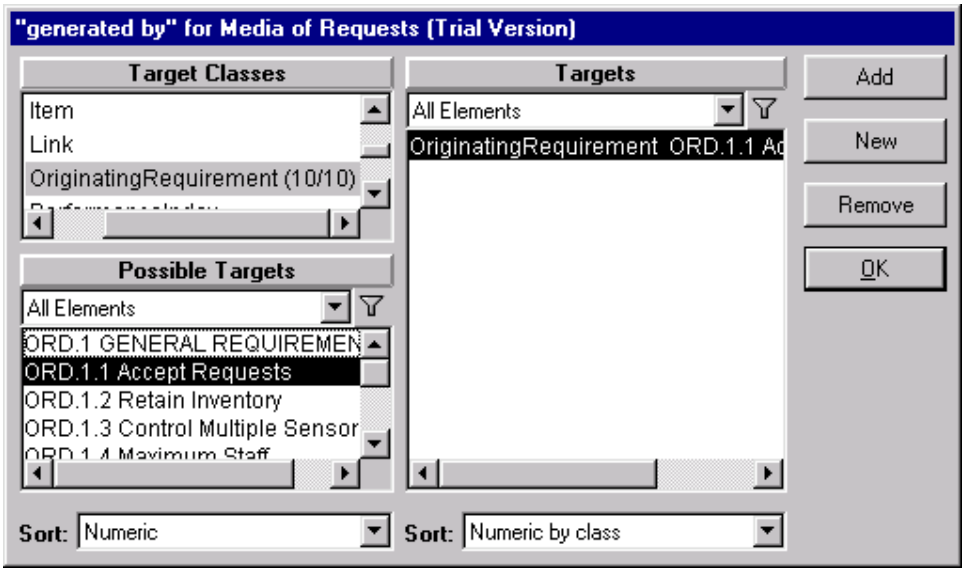
- Close any open windows, except for the **Database Browser** and **CORE Control Panel**.
- From the **Database Browser**, double-click **Issue** in the *Classes* pane to create an *Issue* element.
- Name the Issue **Media of Requests** and press **OK** to close the dialog.
- From the **Database Browser**, double-click the new **Media of Requests** element to open a Text View window.



We link an *Issue* to the element that generated the problem via the generated by relationship.

- Double-click generated by in the *Relationships* pane.
- Add the **Originating Requirement, ORD.1.1**, **Accept Requests** as the target.
- Click **OK**.



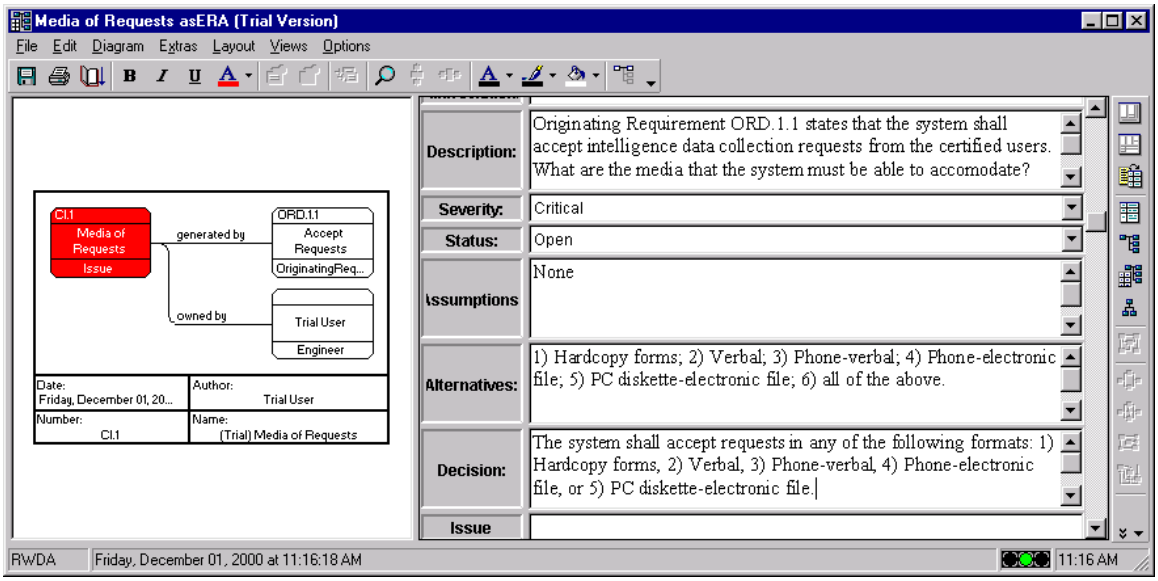


Enhancing the System Definition: Issues (cont.)

CORE allows you to capture both the issue and the decisions that resolve the issue. You can document your decision, your alternatives, and your rationale. In this way, **CORE** serves as a repository for the project design history- capturing the why of system engineering decisions. We will complete the **Media of Requests** Issue element to capture the analysis of the problem and its resolution. Although you can use the **Text View** to fill in the attributes, let's use and **ERA View** this time. With an **ERA View**, you can see a diagram of the *relationships* and a **Text View** of the *attributes* of the selected element.

- From the **Database Browser**, select the **Media of Requests** Issue element.
- Click the **ERA** button to open an **ERA View** of the selected element.

ERA Diagram



- Type text in the *Description* and *Alternatives* fields. You can use the text as shown above, or come up with your own.

Scroll through the list of Issue attributes to see the other fields applicable to an Issue. As a System Engineer, it is important to be as thorough as possible. **CORE** provides the fields you will need to maintain complete requirements.

- Close all **CORE** windows except the **CORE Control Panel**.

This Page Intentionally Left Blank.

Defining the System

In this section, you define your system

- **Create System Elements**
- **Define System Boundaries**
- **View Physical Hierarchy**
- **Define High Level Functions**

Defining the System and its Boundaries

For defining the system boundary, we want to identify the top-level components (physical elements) their top-level (root) functions and any top-level inputs and outputs. To begin, we need to define the overall system context to determine what is inside and outside of our system. The *System* element is intended to identify the system and capture the system-level mission, I/O, functions, performance, and components. We will create the System element with the **ELEMENT EXTRACTOR** because we can get the text from the *CollMgt.txt* file and save some typing.

Open an Extractor Window:

- From the **CORE Control Panel**, click the **ELEMENT EXTRACTOR** button.
- Navigate to and load **Samples/Collmgt.txt** and double-click (or click **OPEN**.)
- From the *Select Element Class* dialog, select the **System** class as the destination class.

Define the attributes for the Element:

- Fill in the **Name**, **Number** (**SYS.1**), **Description**, and **Mission** fields.

Next, we define a relationship by establishing that the System Element is documented by the Document Element, which we had named *Collection Management System*.

Establish a System Relationship:

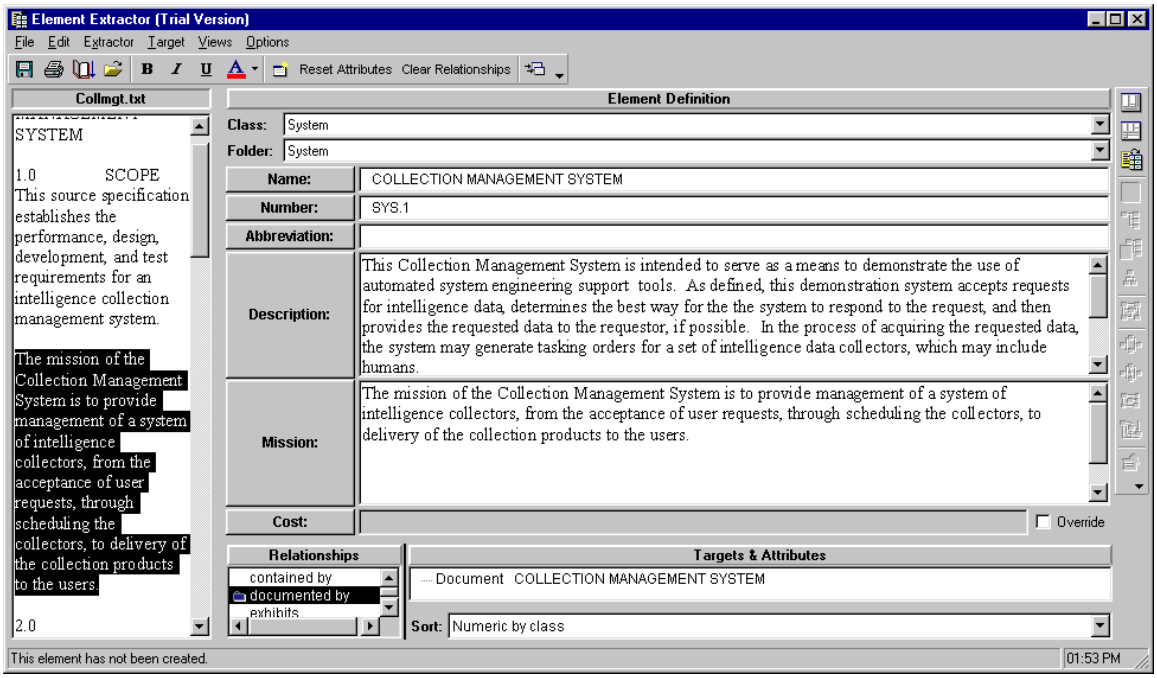
- Double-click the documented by relationship to open the **Target Dialog**
- Select **Document** from the *Target Classes* pane.
- Double-click **Collection Management System** from the list of *Possible Targets* to add it to the *Targets* pane.
- Click **OK** to close the Target Dialog.

Save the Element Attributes in the Database:

- From the **Element Extractor** window, click the **CREATE** button to store this *System Element* definition.

Close the Element Extractor window:

- From the **Element Extractor** window, click the ⊠, or choose **File > Close Window** to close the **Element Extractor** window.



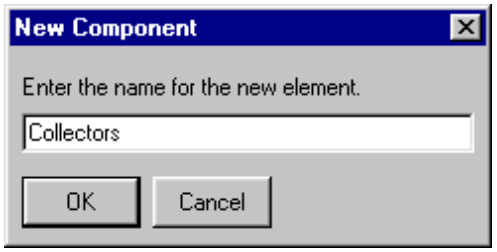
Creating External Systems

External systems interact with the System element. External systems are represented in the system description database by the *Component* class. A *Component* is an abstract term that represents the physical or logical element that performs a specific function or functions. We will create two physical Component class elements, *Collectors* and *Customers*. We will also create a logical Component element called *Universe* to provide a context as to how our system interacts with other systems to achieve its objective.

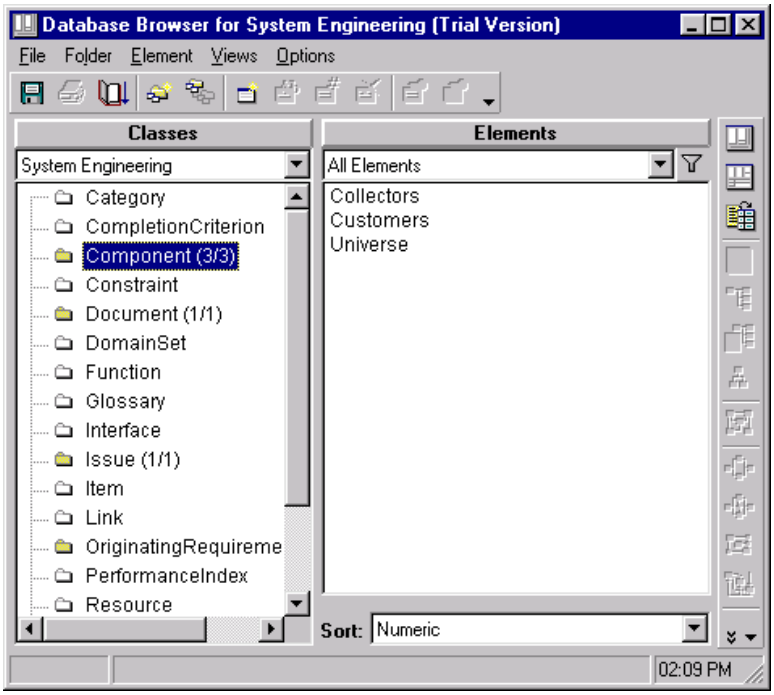
We will use the **Database Browser** to create these elements since this data does not come from the source document--e.g.; there is no data to get from the **Element Extractor**.

Create three new Component Elements:

- From the **CORE Control Panel**, open a **Database Browser** window.
- Double-click **Component** from the list of *Classes* to create a *Component* element. The *New Component* dialog prompts for the element name.
- Name the element **Collectors** and press the Enter key (or click OK.)



- Double-click **Component** again to create an element named **Customers**.
- Double-click **Component** again to create an element named **Universe**.



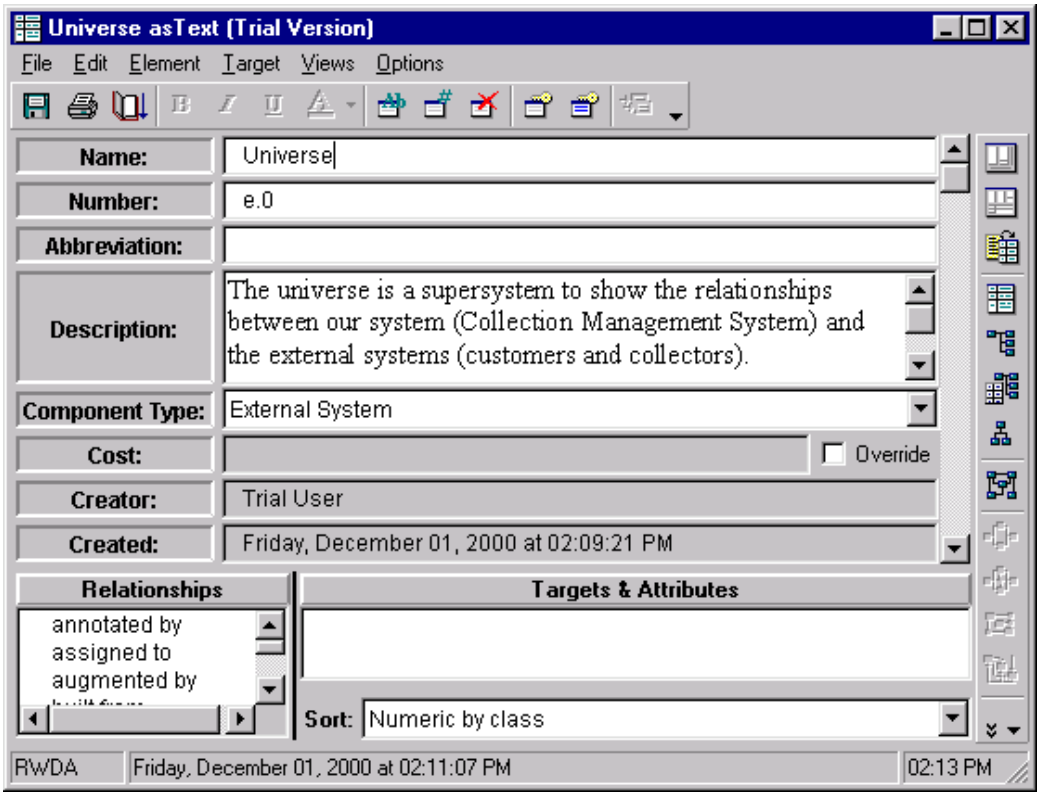
Note
Pressing the Insert key also allows the creation of a new element.

Defining the Universe Component

Now we need to define the attributes and relationships of these elements. We capture the external systems as Components with *ComponentType* set to *External System*. Hierarchy of Components can be constructed using the built from relationship. We want to show that the *Universe* is built from the *Collection Management System* and the two external systems, *Collectors* and *Customers*.

Define Attributes and Relationships:

- Open a **Text View** by double-clicking **Universe**.
- Type the attribute information as shown below.

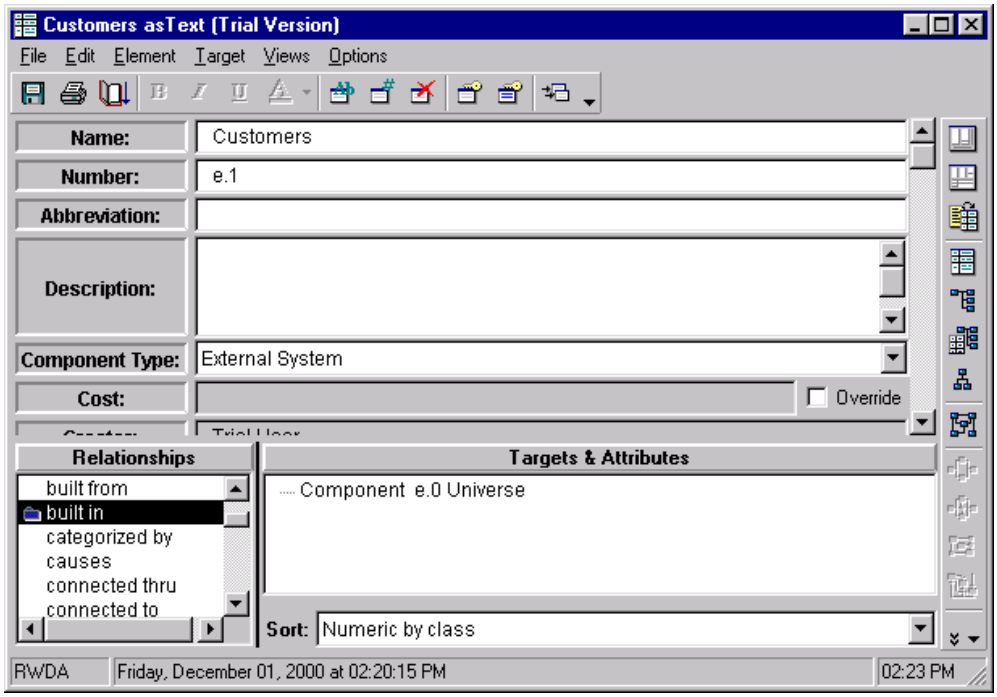


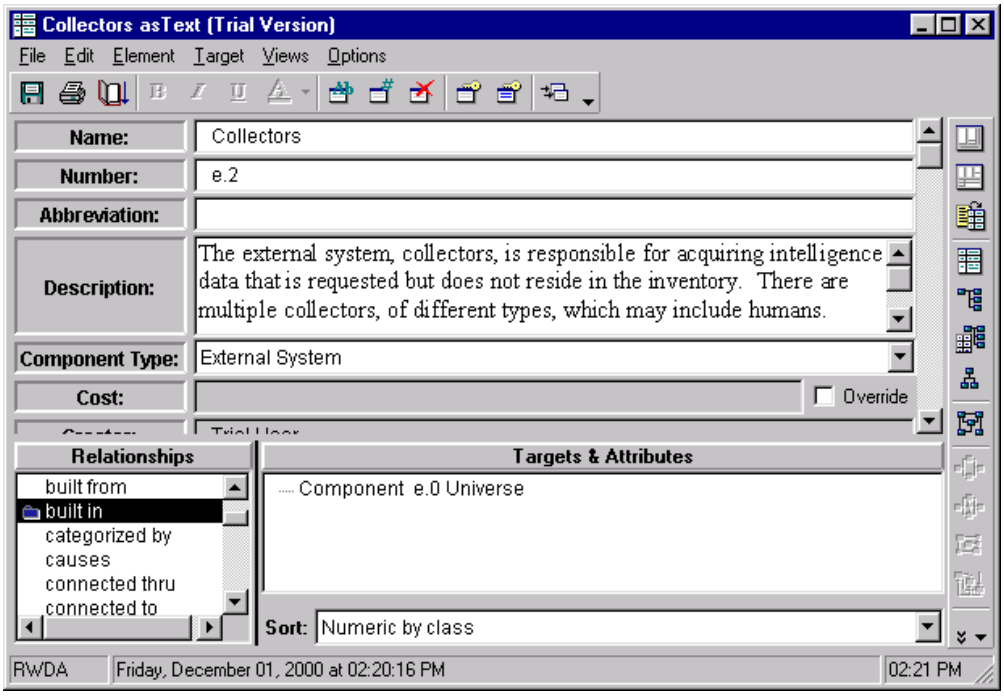
Adding Detail to the External Systems

(Continuing)

To be a thorough System Engineer, you will want to complete the attribute fields for the Collectors and Customers Component Elements.

- From the **Database Browser**, open a **Text View** for each of these Component Elements
- Fill in values for the *Number* and *Description*, and set the Component Type to **External System** from the pull-down list.
- Close all the **Text View** windows when you are done.

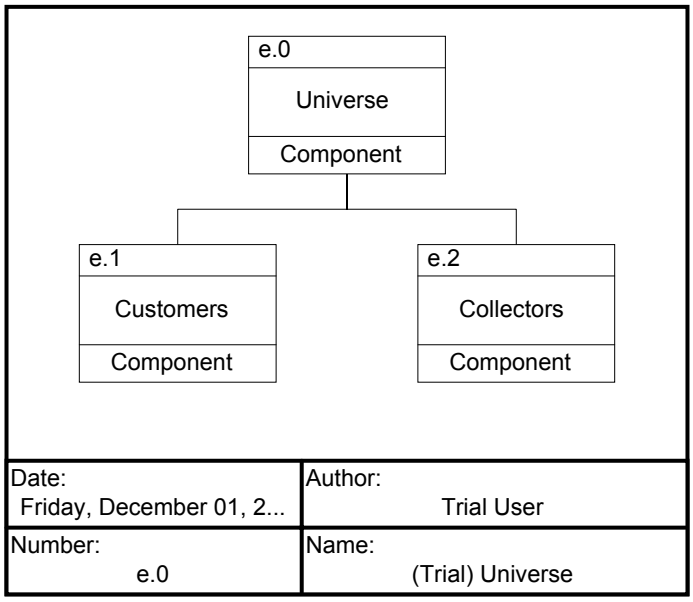




Viewing a Physical Hierarchy

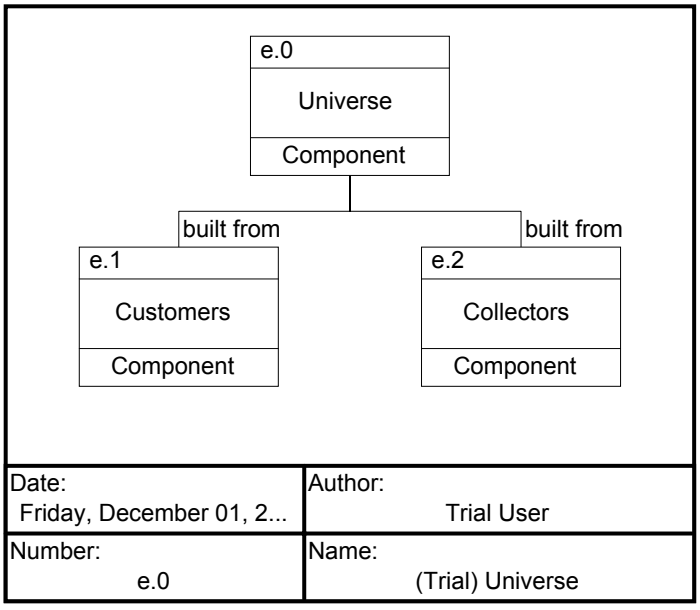
To see the structure of the **Universe Component Element**, we can view a **Physical Hierarchy** diagram.

- From the **Database Browser**, highlight the Component **Universe** element.
- Click the **Hierarchy** button.
- Select **Physical** from the enumerated list of stored definitions.
- Click **OK**.



You will get a **Physical Hierarchy** diagram representing the context of our system.

- Choose **Options > Show Relationships** to annotate the diagram.
- Close all open windows, except for the **CORE Control Panel**.

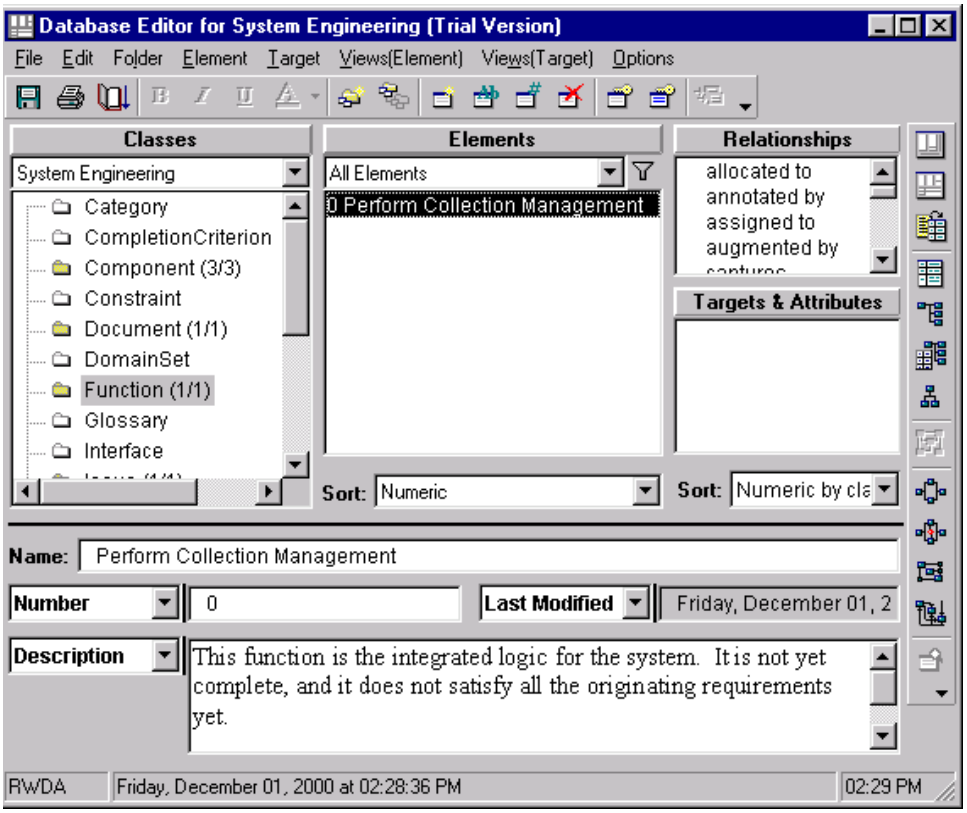


Creating Function Elements

We have defined our *Document*, our *Originating Requirements*, our *System*, and some *Components*. Now we need to add some *Functions* to describe what our system is to do. A *Function* is a transformation that accepts one or more inputs and transforms them into outputs. Let us create the root (top-level) function for our system and each of the external systems. With any system, a root-level function drives the other functions. This time we will use the **Database Editor** to create these elements. The **Database Editor** allows us to create elements and define the attributes in the same window. It is good practice to fill in the applicable attributes of elements as you create them, especially the *Description* since it represents the requirement statement for the element.

To Create Function Elements:

- From the **CORE Control Panel**, click the **DATABASE EDITOR** button.
- In the **Classes** pane, double-click **Function** to open the *New Function* dialog.
- Type the name **Perform Collection Management**
- Type a description for this function in the *Description* field.



Create three additional *Functions* named as follows:

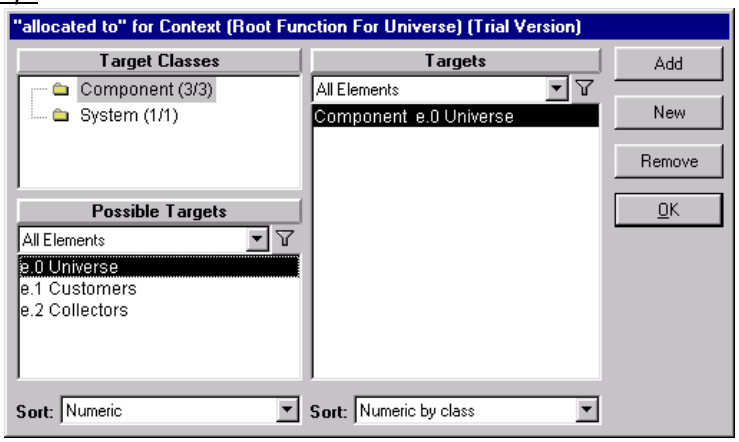
- **Context (Root Function For Universe)** number **u.1**,
- **Perform Collector Functions** number **c.1**, and
- **Perform Customer Functions** number **c.2**.

Create Root Functions for the System/External Systems

We will now allocate the root functions to the components that perform them. This means establishing an *allocated to* relationship.

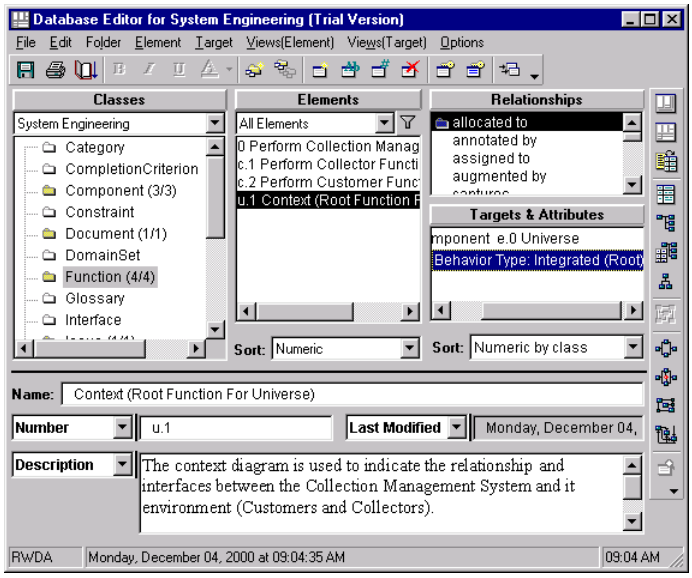
Establish an allocated to Relationship:

- From the **Database Editor** highlight the *Function Element* **Context (Root Function For Universe).**
- Double-click the relationship, **allocated to** to open a **Target Dialog**.
- Highlight the target class *Component*, and then double-click the **Possible Target**, **E.0 Universe** to add it to the *Targets* pane.
- Click the **OK** button to close the **Target Dialog**.



- Double click on **Behavior Type**: **Atomic**.
- From the **Edit Behavior Type Attribute** dialog, select **Integrated (Root)** from the enumerated list as the new *Behavior Type* and click **OK**.

You are now finished allocating the Context function.



Allocate the other two external root functions (**Perform Collector Functions** and **Perform Customer Functions**) to their respective Component elements the same way.

Also, allocate **Perform Collection Management** to the System element and identify its *Behavior Type as Integrated (Root)*.

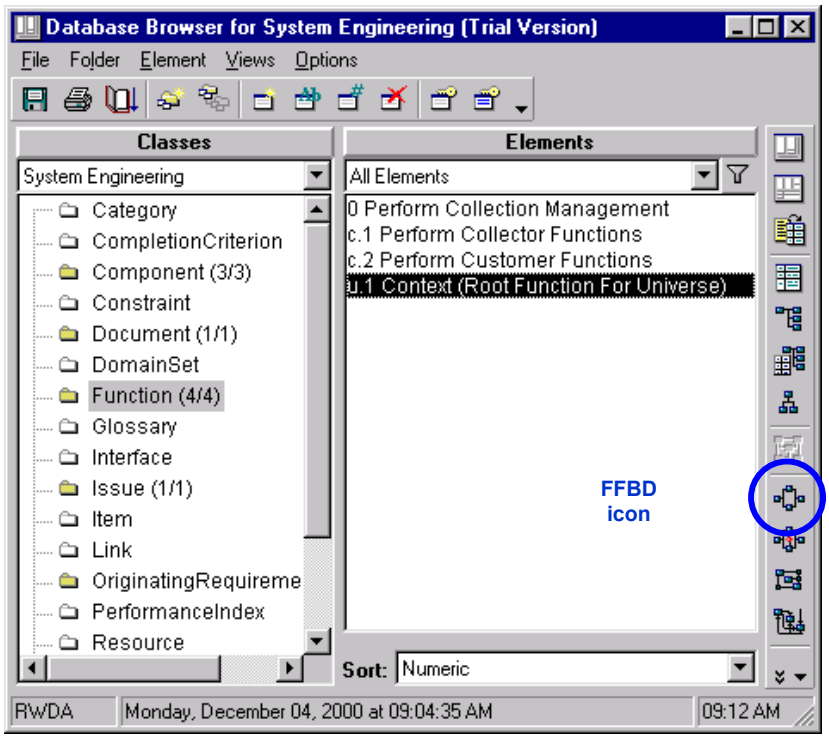
Note
Make sure the model has a System and a corresponding root Function (behavior type Integrated Root) for the system being developed. Even during requirements analysis, this is important as it provides a standard starting point for analysis and review.

After finishing, close the Database Editor.

Building a Functional Model

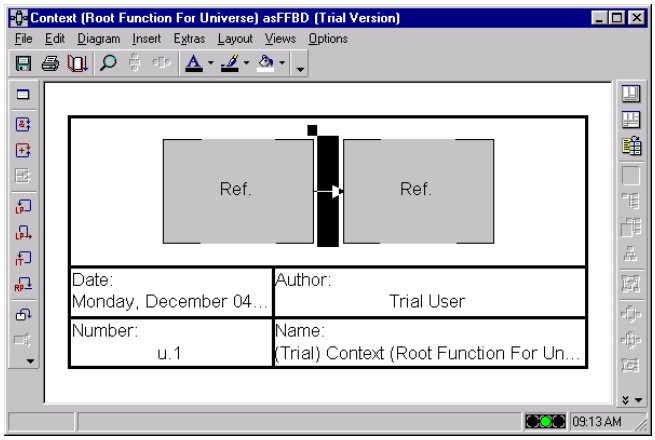
Next, we begin the behavior analysis of our system. We can do this graphically with a functional model of our context function, **Context(Root Function For Universe).**

Open Database Browser



Open an FFBD:

- Highlight the **Function** Element, **Context (Root Function For Universe).**
- Click the **FFBD** icon to open a functional flow block diagram of the selected element.
- Select/highlight the branch between the two reference blocks. This is where we will insert the functional behavior for the context function, the root function for Universe.

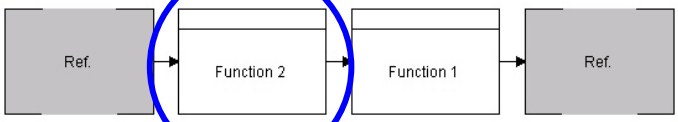


A Note About Insertion Points and Selecting Objects and Branches

Selecting An Object

Selecting the object on logic constructs causes the insertion to be to the *left* of the selection.

Result

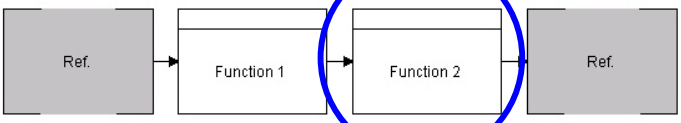


New object is inserted to left

Selecting A Branch

Selecting the branch on logic construct causes the insertion to be to the *right* of the selection.

Result



New object is inserted to right

Inserting a Parallel Structure

Recall, we have three functions that we want to include in our diagram:

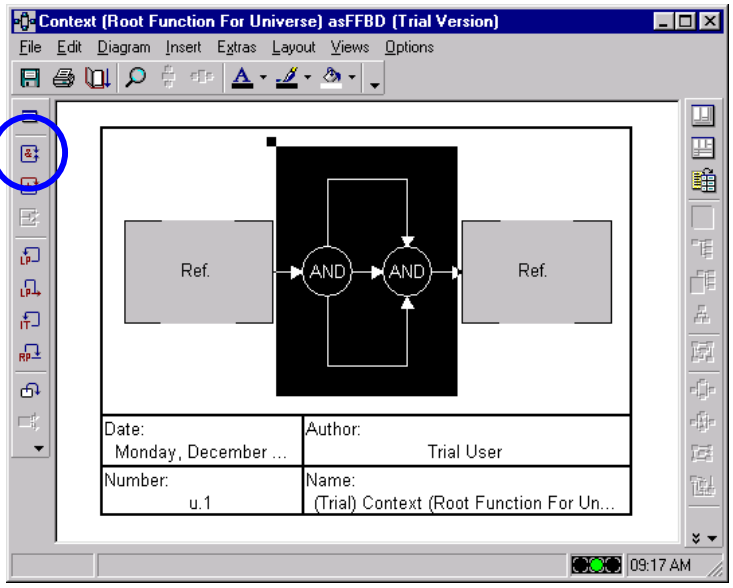
- **Perform Collection Management**,
- **Perform Collector Functions**, and
- **Perform Customer Functions**.

Each of these functions works in parallel. To incorporate these functions into our diagram, we will create parallel branches.

Creating a Parallel Diagram Structure:

- From the **FFBD** window, choose **Insert > Parallel** menu command to insert a concurrency structure.

(You may also use the Structure icon.)



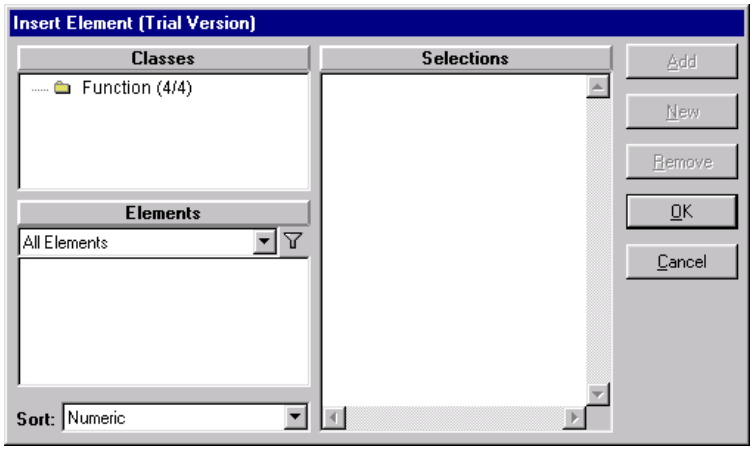
CORE prompts for the number of branches. We will create a concurrency with 3 branches since we have 3 functions.

- Type **3**, then **OK**

Now we insert one function on each branch. To insert an element on a branch, you click on the branch to highlight it, and you select the menu commands **Diagram > Insert > Element**.

Adding Functions to a Diagram:

- Highlight the top branch.
- In the **FFBD** window, choose **Insert > Element** command to open an **Insert Element** dialog.

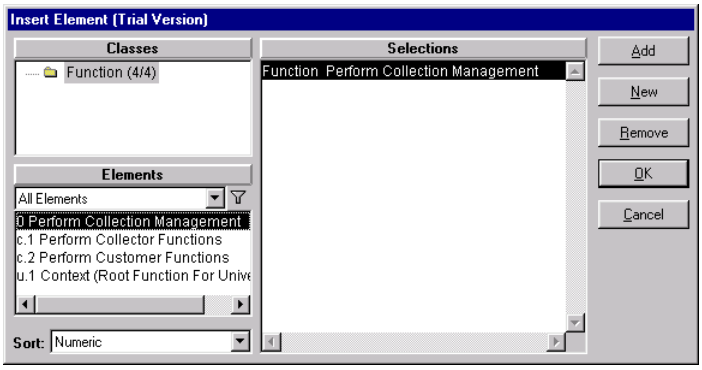


Building a Functional Model

The **Insert Element** dialog works like the Target Dialog; it shows elements that can be inserted. You can select one of those to add to the Selections pane, or you can create a new element by clicking the **NEW** button. Click **OK** and all the elements listed in the Selections pane are added to the highlighted spot in the diagram, in the order they are listed in the Selections pane.

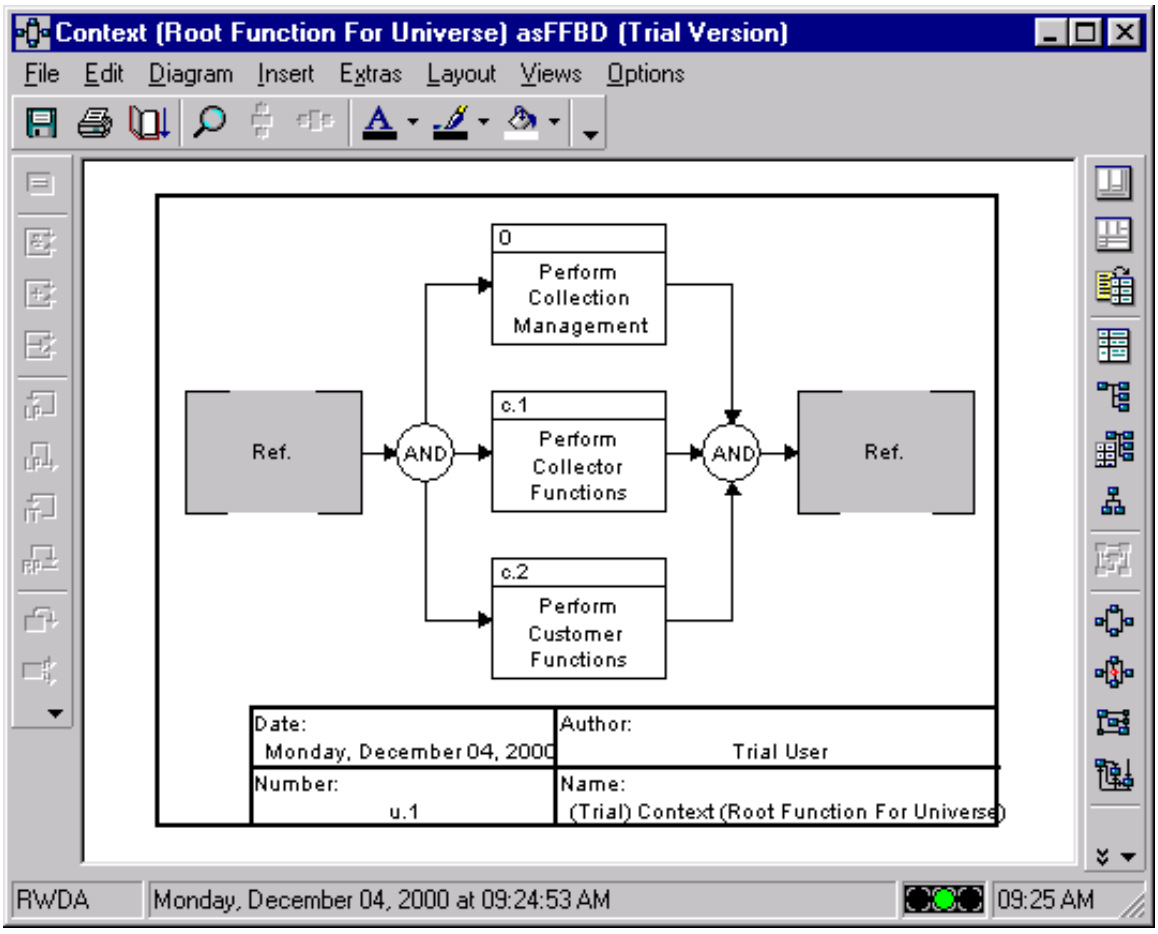
Inserting Elements:

- From the list of *Classes*, highlight **Function**
- From the list of *Elements*, highlight **Perform Collection Management**.
- Click **ADD** to add the element to the *Selections* pane.
- Click **OK**.



The Function Element is added to our **FFBD**.

In the same manner, insert **Perform Customer Functions** on the second branch and **Perform Collector Functions** on the third branch.

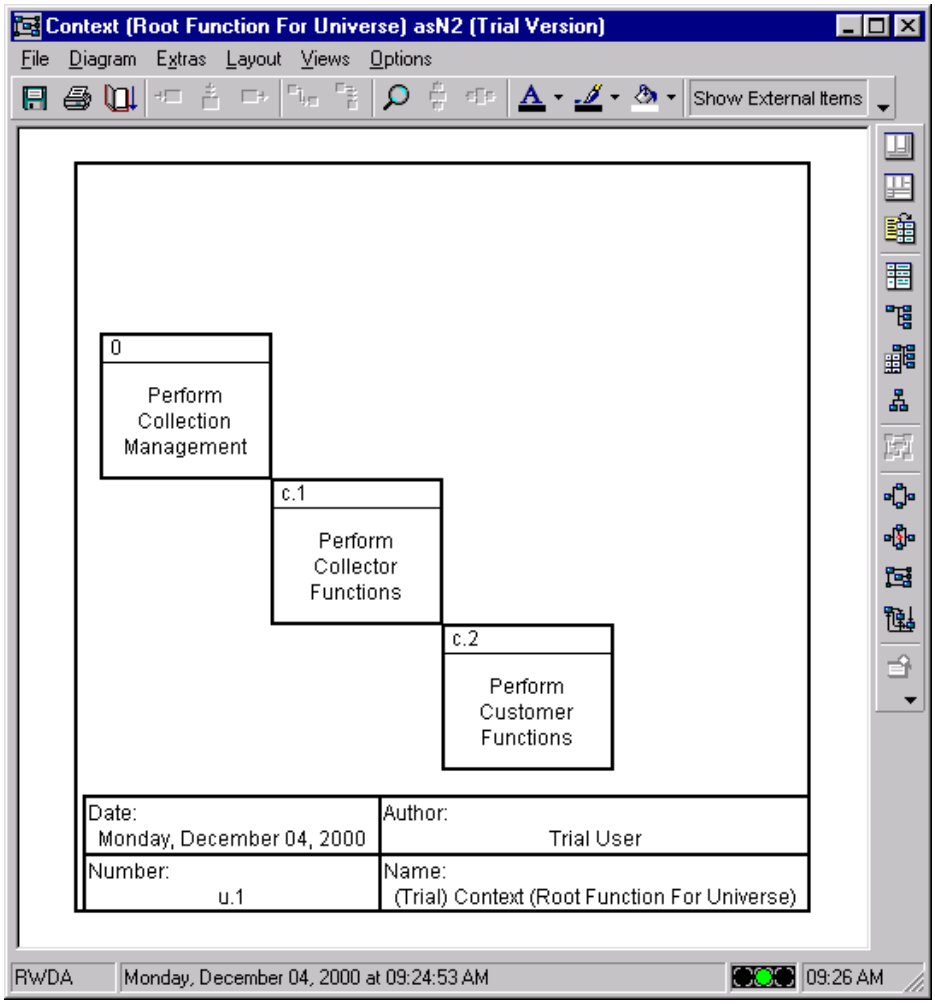


Adding Inputs and Outputs

Now let's add inputs and outputs to the root function (and thereby to the Components that they are allocated to) to show the activities of the Functions. We will do this by using the **N2** (interface) view. In a later exercise, we will show how to add data flows from the **EFFBD** view.

Open an N2 Diagram:

- Click on the diagram background to ensure no icons are selected. We don't want any icons or branches selected.
- Choose **Views > N2** to open an **N2 Diagram** of this function, Context (**Root Function For Universe**).



An **N2** *Diagram* represents the data flow for a system or system segment. It displays the data dimension of the behavior model, whereas the **FFBD** displays the control dimensions of the integrated behavior model. Used in conjunction with an **FFBD**, you can use the **N2** diagram to help capture and analyze the functional behavior of a system.

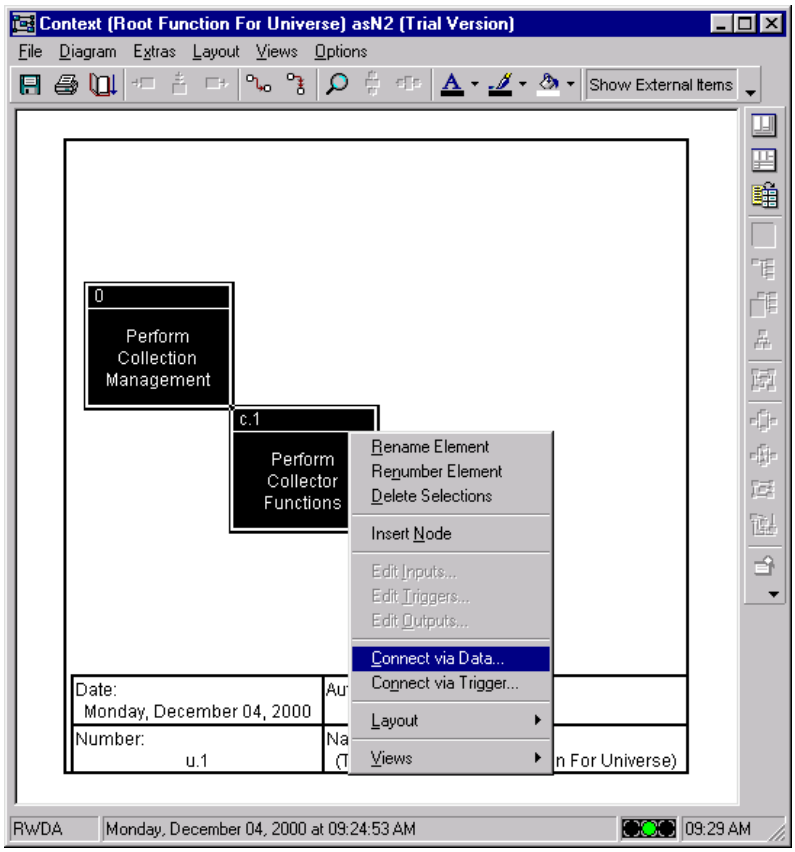
Note
Editing the diagrams is another way of entering data into the CORE repository.

Adding Inputs and Outputs (cont.)

Item Elements represent flows with and between Functions Elements. An *Item* is an input to or an output from a Function. We will create an Item Element named **Tasking**. Tasking will be an *output from* **Perform Collection Management** and an *input to* **Perform Collector Functions**.

Opening a Connection Dialog:

- On the **N2** Diagram, click to select the icon **Perform Collection Management.**
- Now, while holding down the shift-key click to select the **Perform Collector Functions** icon.
- Right-click to open a pop-up menu and choose the **Connect via Data...** command to open a **Connection Dialog**.

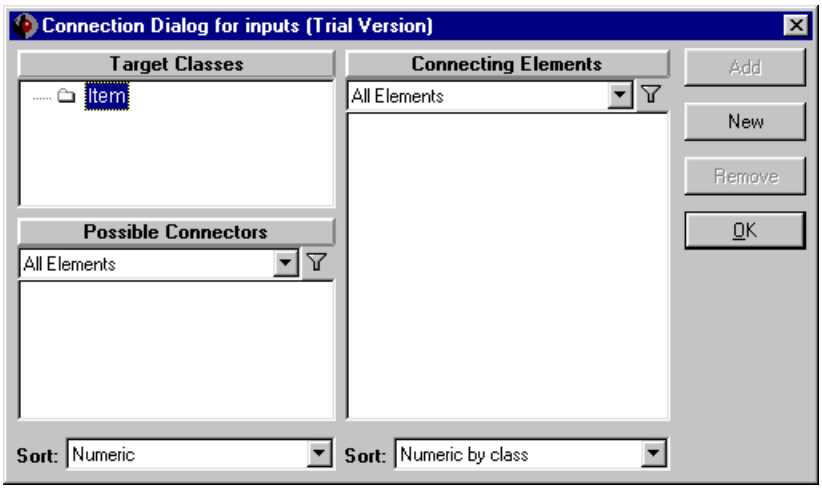


Adding Inputs and Outputs (cont.)

Since the Item Element named **tasking** does not exist, you must create it.

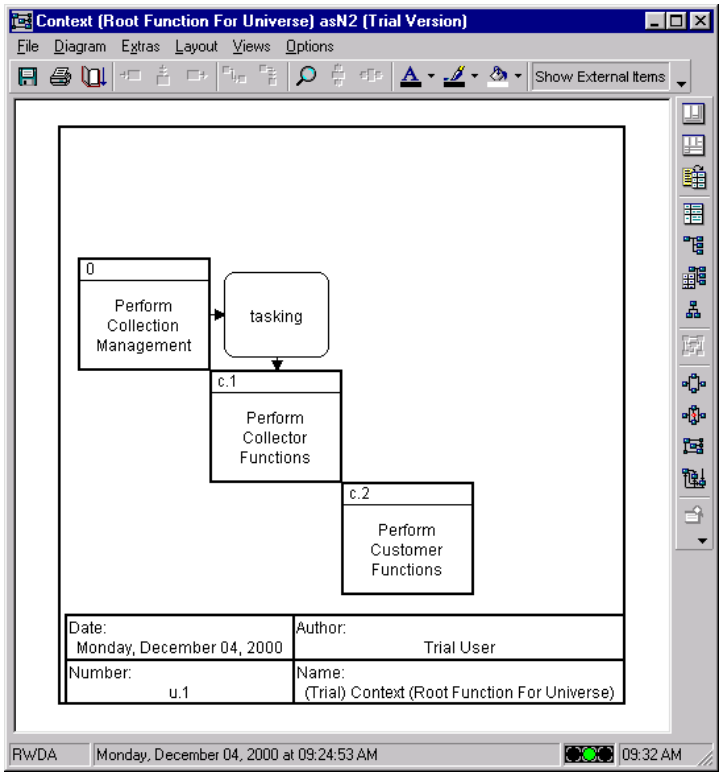
Creating an Item Element:

- Highlight **Item** from the list of **Target Classes**.
- Click **New**.



- Type **tasking** for the name of the created **Item Element**.
- Click **OK** to close the *Connection Dialog*.

The *tasking Item* is added to the **N2** Diagram. Notice the direction of the arrow; the tasking item is an output of **Perform Collection Management** and an input to **Perform Collector Functions**.



The N2 Chart Shows the Interfaces for Our System

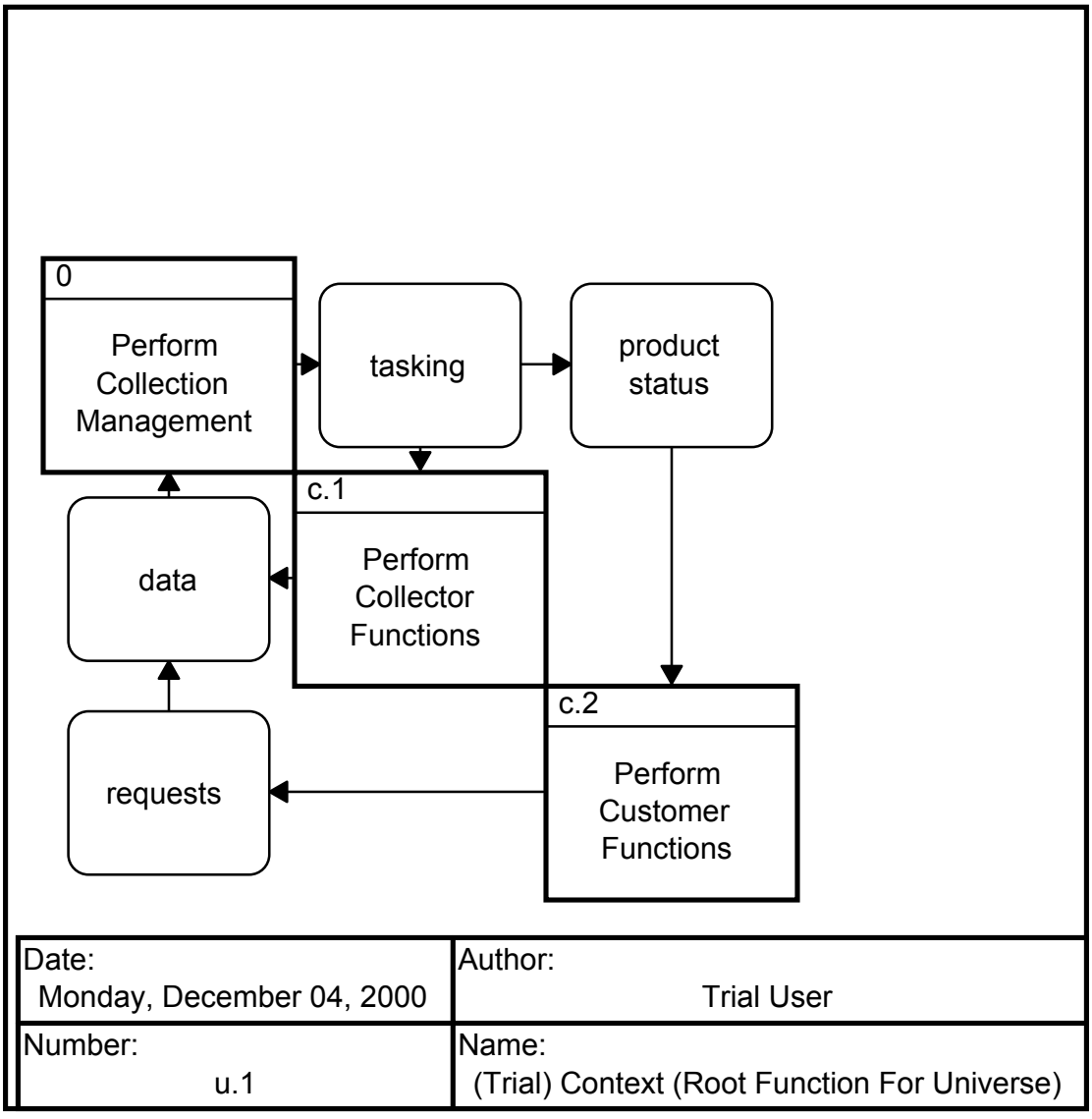
We'll now add three additional data items in the same manner.

Using the chart as a guide for the desired information flow between our system and the two external systems, complete the model for the three other items: *product status*, *requests*, and *data*.

Context (Root Function For Universe) Interfaces

Source (Output)	Input	Item
Perform Collection Management	Perform Collector Functions	tasking
Perform Collection Management	Perform Customer Function	product status
Perform Customer Functions	Perform Collection Management	requests
Perform Collector Functions	Perform Collection Management	data

Below is the completed **N2** view showing the interfaces (Item flows) between the three functions.

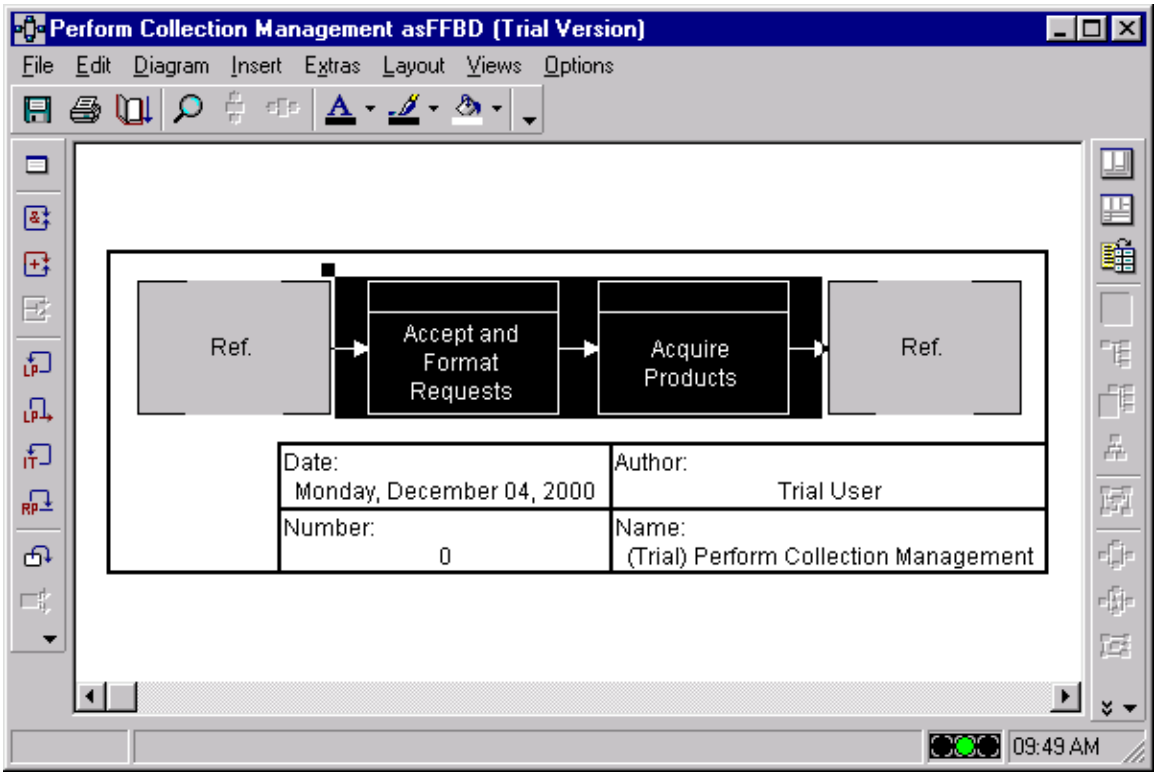


When you are done, close the diagram window.

Deriving the Functional (Behavior) Model for Our System

Now we will decompose the Functions, breaking them down into leaf-level Functions. We begin with the Function **Perform Collection Management**.

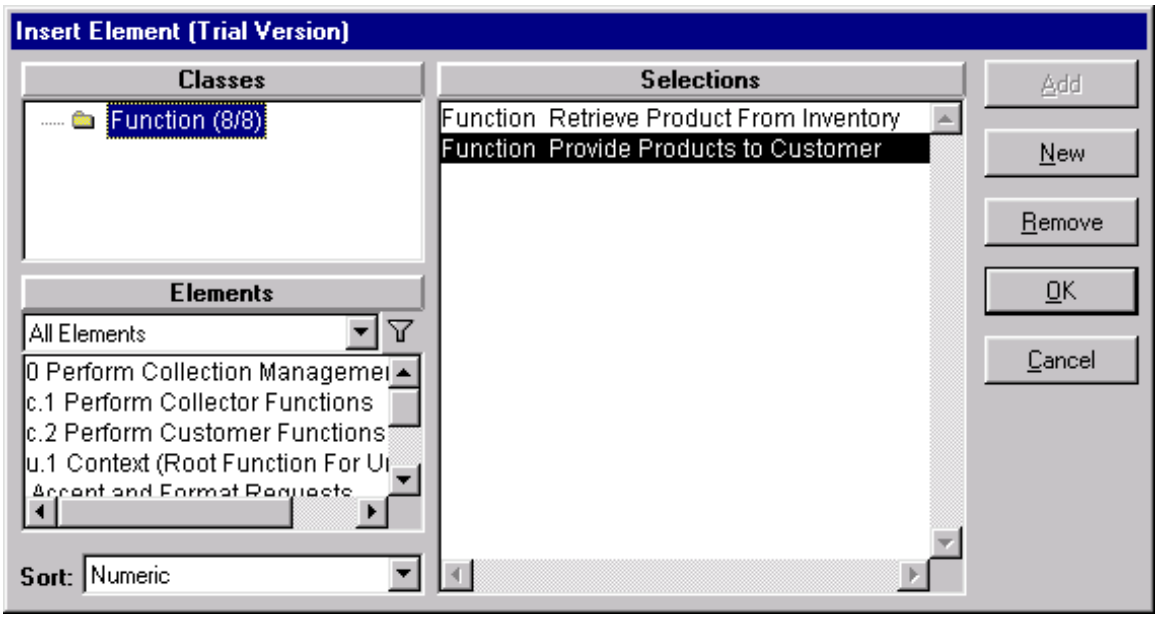
- From the **Database Browser**, select the Function Element **Perform Collection Management.**
- Click **FFBD** to open an **FFBD** view.
- Select/highlight the branch between the reference blocks at the insertion point.
- From the menus, choose **Insert > Element**.
- From the Insert Element dialog, create two new Functions: **Accept and Format Request** and **Acquire Products**.
- Click **OK** to add these functions to the diagram serially in the order they were added to the target list.



Deriving the Functional (Behavior) Model for Our System (cont.)

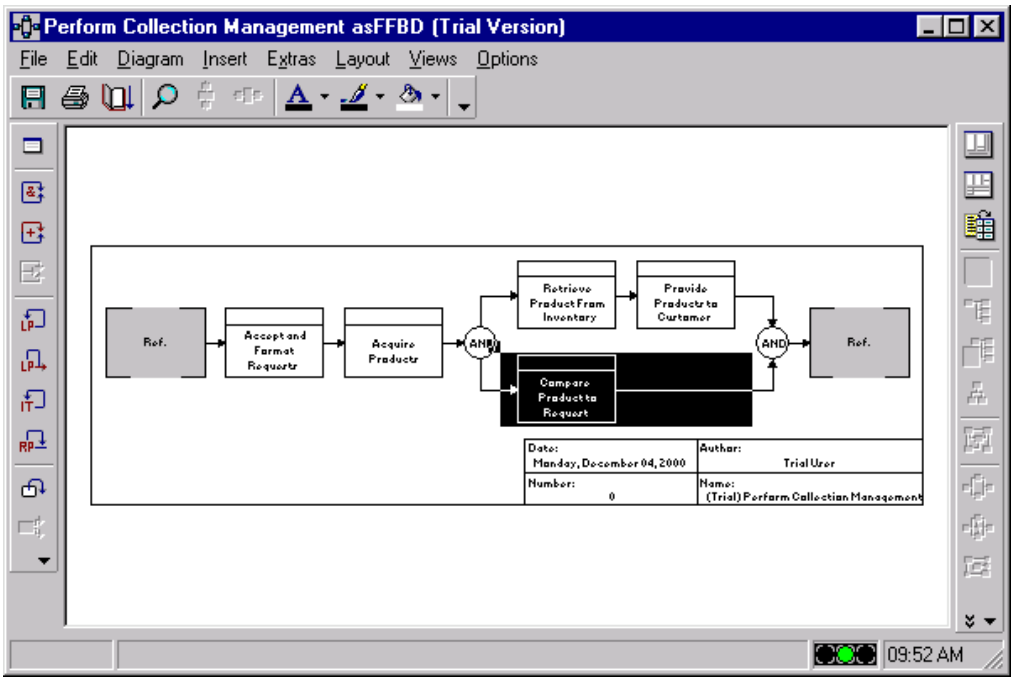
Now while the branch is still selected, from the Diagram menu choose the **Insert > Parallel** command and accept the default of **2** branches. Then, select/highlight the top branch of the concurrency (the parallel construct). From the Diagram menu, select the **Insert > Element** command.

When **CORE** opens the dialog box, create two functions: **Retrieve Product From Inventory** and **Provide Products to Customer**. Click the **OK** button.



Select/highlight the bottom branch of the concurrency. From the Diagram menu, select the **Insert > Element** command.

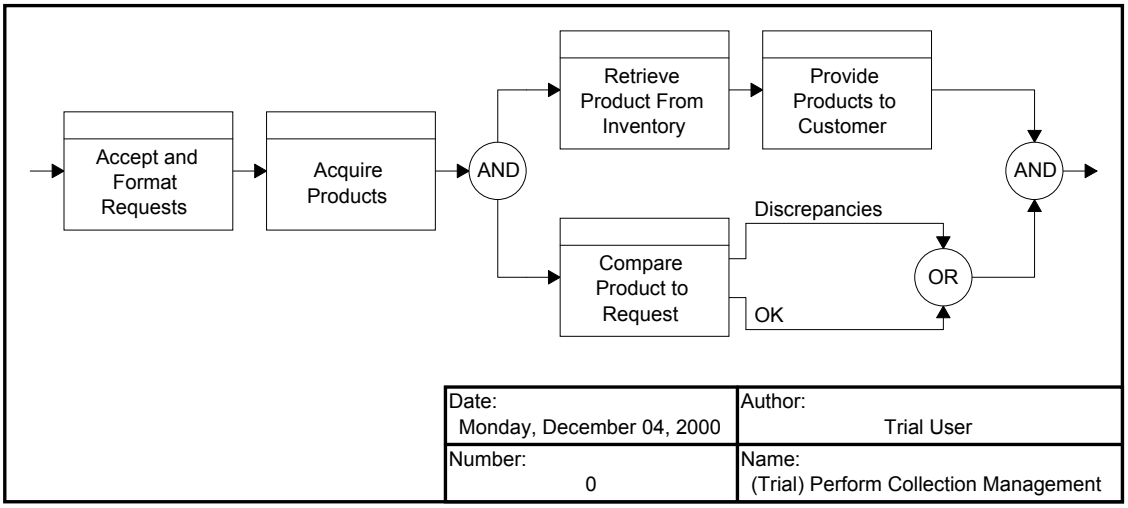
When **CORE** opens the dialog box, create one function: **Compare Product To Request**. Click the **OK** button.



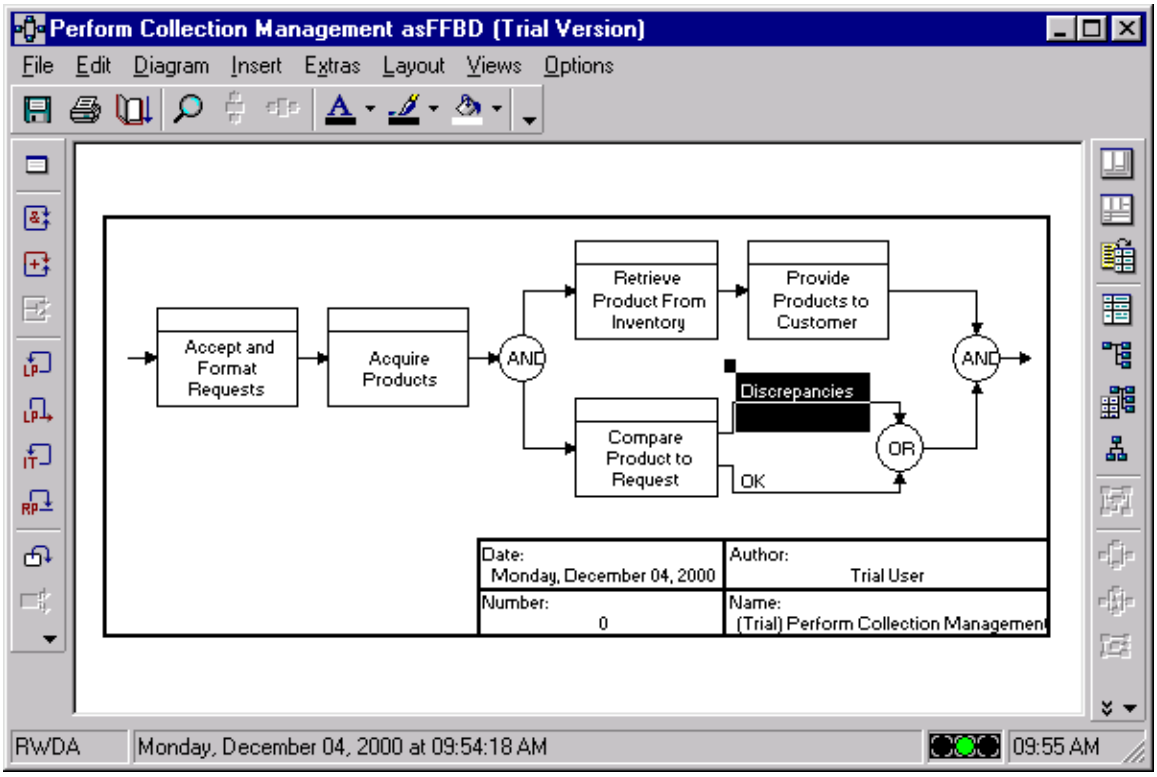
Deriving the Functional (Behavior) Model for Our System (cont.)

To define this function as having multiple exit conditions (paths), select/highlight the function **Compare Product To Request**. Select the **Insert >> Add Exit Condition** command from the **Diagram** menu.

- When **CORE** opens the dialog box, select **Completion Criteria** from the *Classes* pane, then click on the **New** button and create two exit conditions (represented as *Completion Criteria*): **OK** and **Discrepancies**.
- Click the **OK** button when finished.

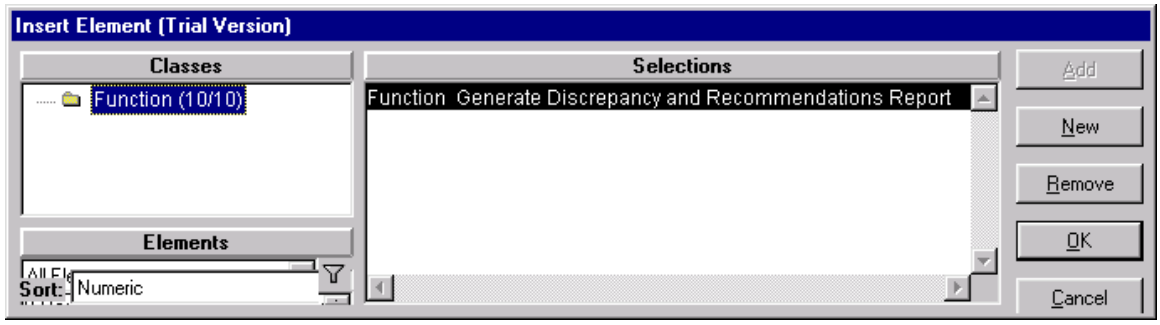


Select/highlight the **Discrepancies** branch of the multiple exit function.

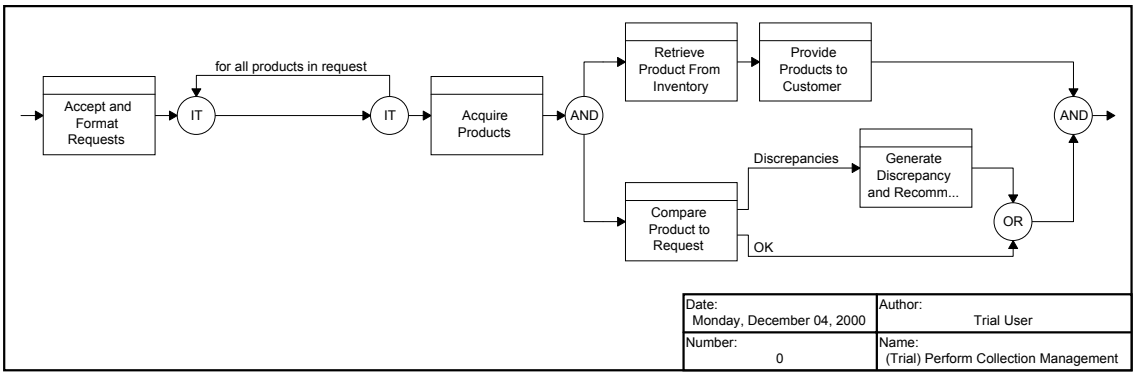


Deriving the Functional (Behavior) Model for Our System (cont.)

From the **Diagram** menu, select the **Insert > Element** command. When **CORE** opens the dialog box, click on the **New** button and create one Function: **Generate Discrepancy and Recommendations Report**. Click the **OK** button.



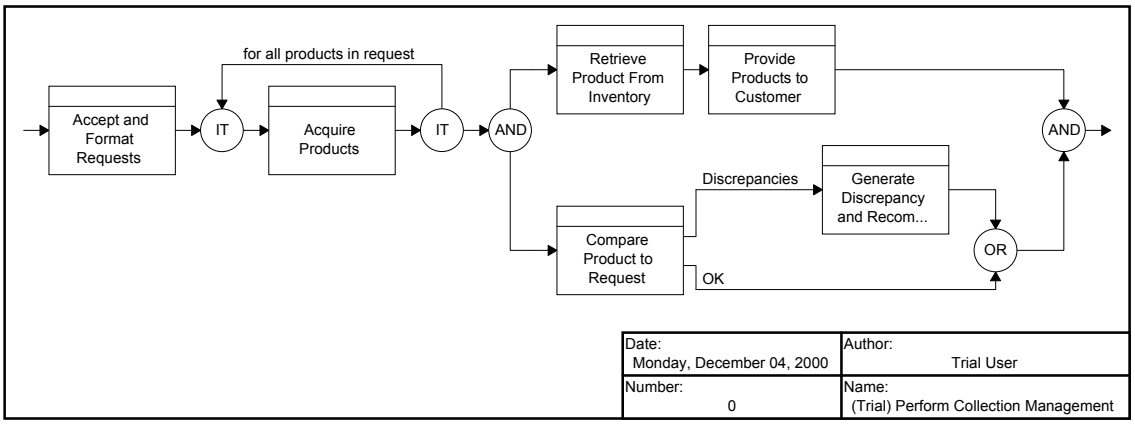
Next, select/highlight the Function **Acquire Product**. From the Diagram menu, select the **Insert > Iterate** command. When **CORE** opens the dialog box, click on the **New** button and create the *DomainSet* **for all products in request**. Click the **OK** button.



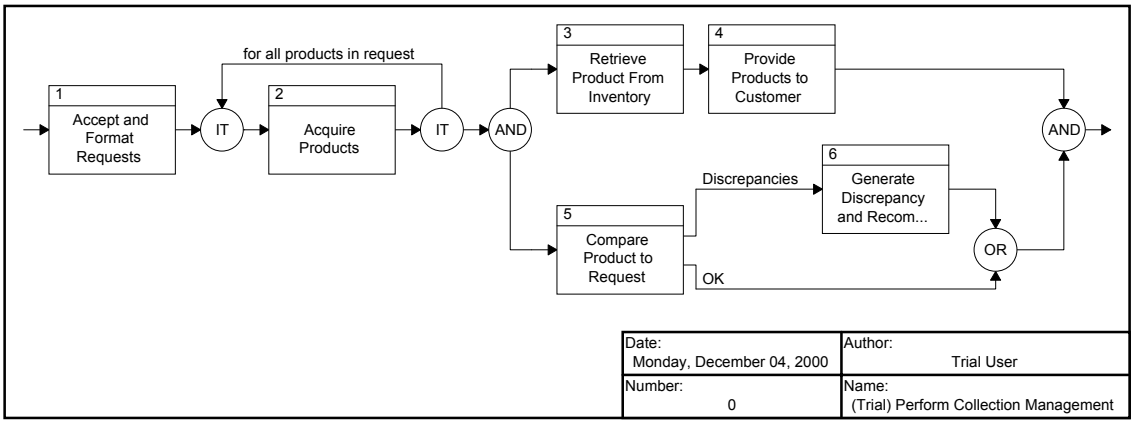
- Ensure that the function **Acquire Product** is selected/highlighted.
- From the **Edit** pull-down menu, select the **Cut** command (or use the Microsoft Windows shortcut **Ctrl+X**).

Deriving the Functional (Behavior) Model for Our System (cont.)

Highlight the main branch inside the Iterate. From the **Edit** menu, select the **Paste** command. This logic assumes that a single request may ask for more than one product.



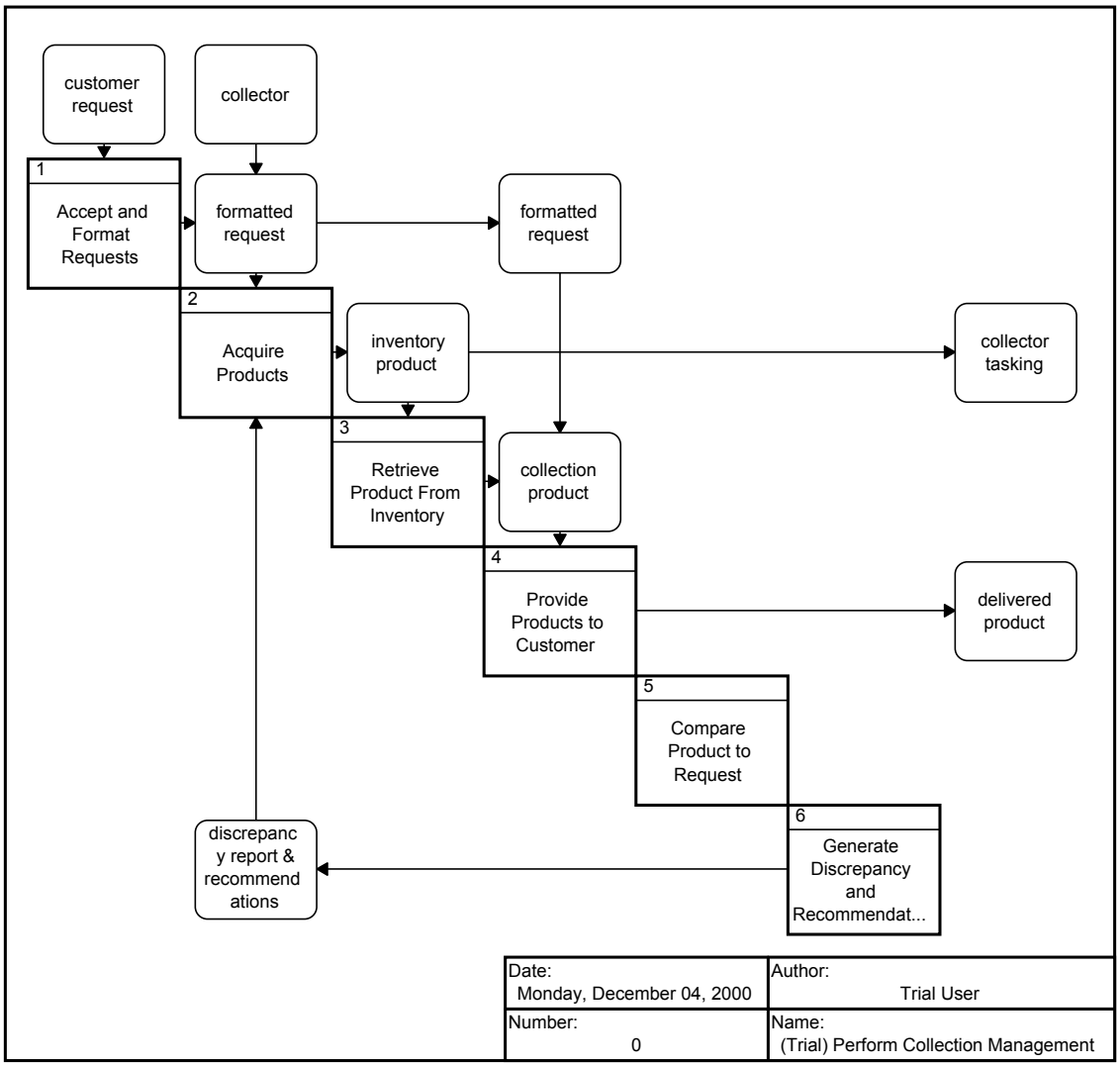
As a final step, let us renumber the functions on our **FFBD**. **Be sure that nothing is selected/highlighted.** From the **Diagram** menu, select the **Renumber Element** command. When **CORE** prompts, enter 0 (zero) as the number for the parent function. Click **YES** at the next warning prompt. This will cause the functions on the diagram to be numbered with integers.



Adding Inputs and Outputs (N2 Charts)

Remembering the N2 Charts:

While we have the **FFBD** open let's define the Items which are inputs and outputs to the functions as shown in the **N2** chart and the table of the **Perform Collection Management** root function. We will do this using an **N2** chart, as we did earlier for the **Context (Root Function for Universe)** function. From the **Views** menu, select the **N2** command.



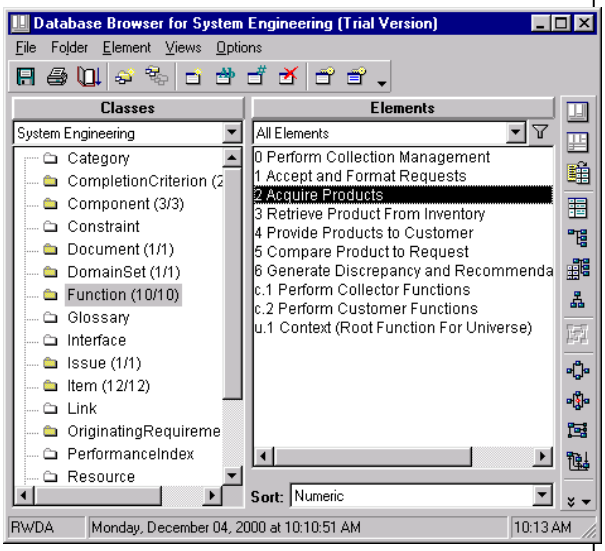
Selecting a single function allows you to define inputs, outputs, and triggers. Selecting multiple functions allows you to connect the function via data or triggers. **Remember to hold down the shift key when selecting from-to connections.**

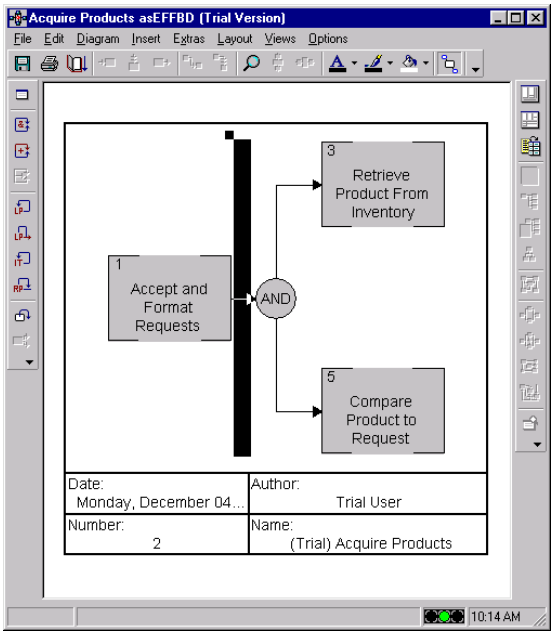
Enhanced FFBD:

Close all windows but the Palette and reopen the Database Browser window. Now, let's decompose Function **2**, **Acquire Products**, to add another level of detail. For this exercise, we will use the **EFFBD** (an **FFBD** with data).

- Select/highlight the Function **Acquire Products**.

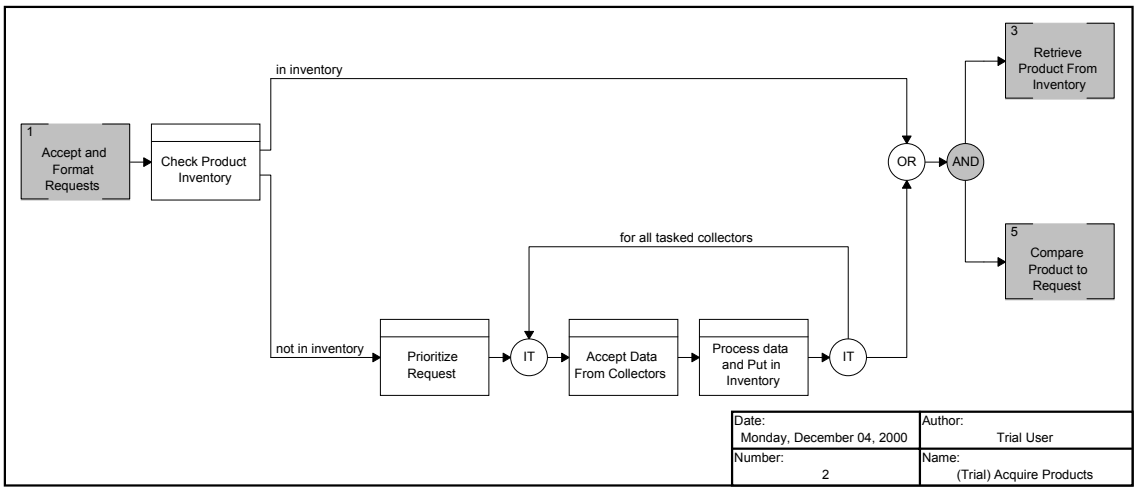
From the **Views** menu, select the **Enhanced FFBD** command to open an **EFFBD** of **Acquire Products**. From the **Database Browser** you can select the function and click the **EFFBD** button.





Highlight the main branch (between the reference block on the left and the concurrency (AND) for the reference blocks on the right).

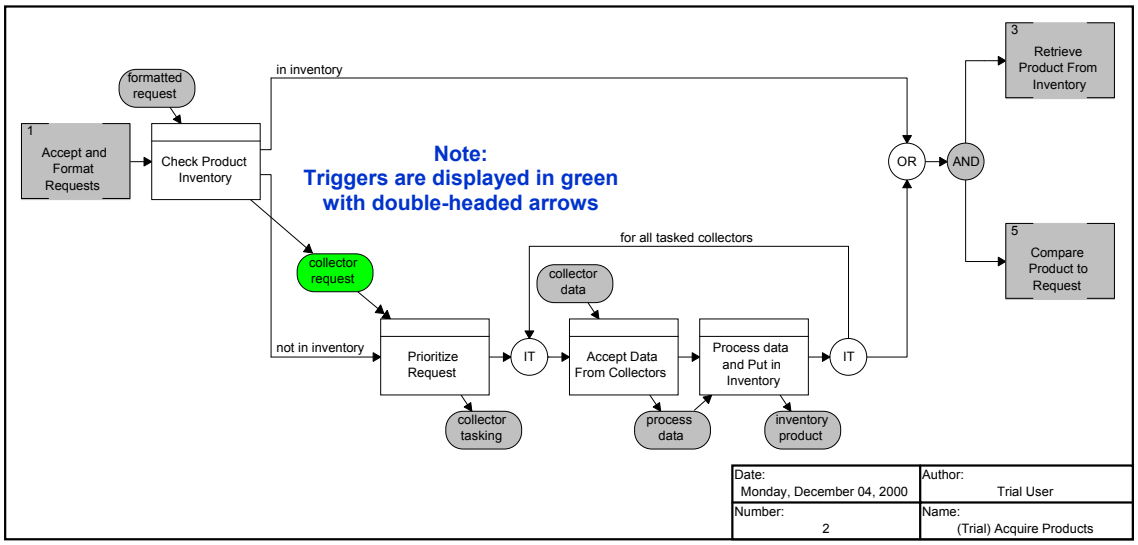
Generate the functions and their control constructs (e.g., the **FFBD** subset of the **EFFBD**) as shown, using the methods presented earlier for decomposing the **Perform Collection Management** function.



Deriving the Enhanced Functional (Behavior) Model (cont.)

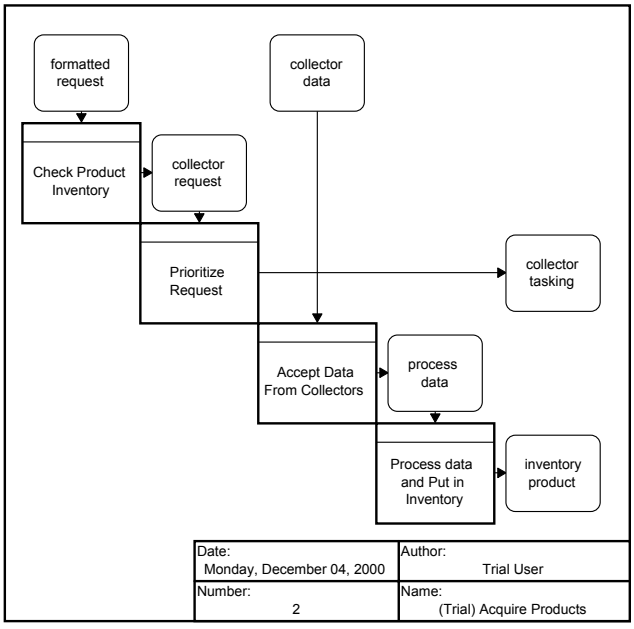
Inputs and outputs can be added to functions in the **EFFBD** in the same manner as we used with the **N2** chart earlier. Flows between functions can be defined by highlighting the functions in order of from-to and using the **Edit > Connect via ...** commands in the Diagram menu.

Individual function inputs and outputs can be added with the **Edit > Inputs** (or **Outputs**, or **Triggers**) command in the **Diagram** menu. Note that triggers differ from data items in that they are required by a Function before it can begin execution.



The completed **EFFBD** and **N2** chart are shown.

Function	Trigger(s)	Input(s)	Output(s)
Check product Inventory		formatted request	collector request
Prioritize Request	collector request		collector tasking
Accept Data From Collectors		collector data	process data
Process Data and Put in Inventory		process data	inventory product



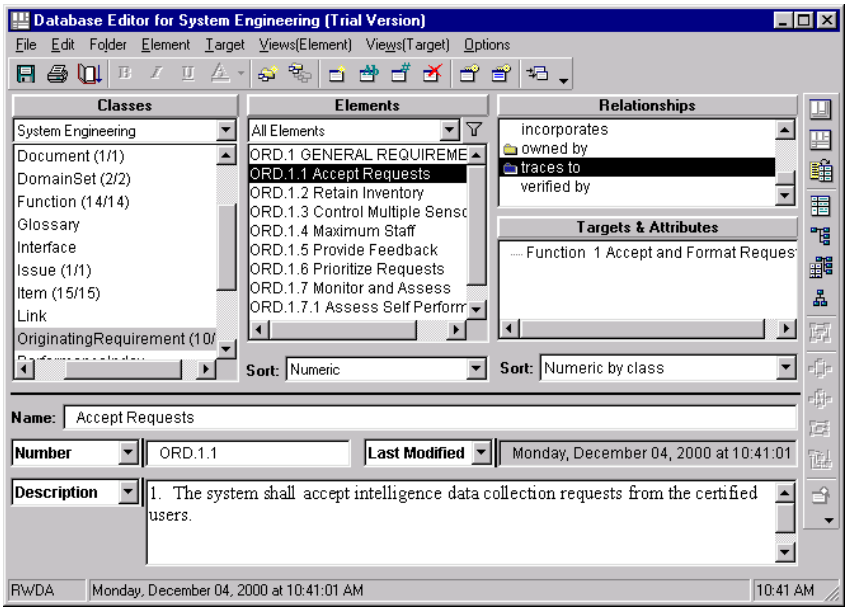
Revisiting/Extending Traceability

Traceability:

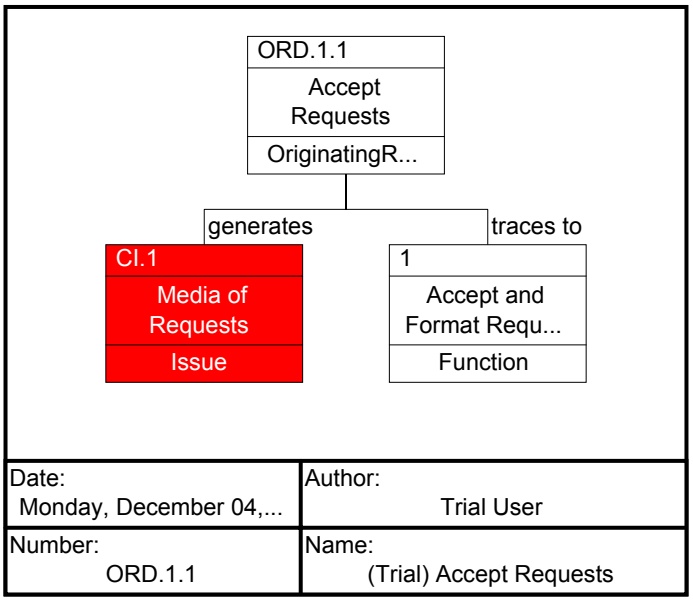
Now that we have defined our functional (behavior) model, let us go back and extend the traceability to include these new elements by simply relating these functions to the requirements they fulfill. We could form these relationships in the **Text** views, the **ER** views, the **ERA** views, or the **Database Editor**. We will use the **Database Editor**.

In the **Database Editor**, in consecutive panes, select class **Originating Requirement**, element **Accept Requests**, and relationship traces to. Then select the **Add Target** command from the *Target* menu or click the **Edit Targets** button.

In the **Target** dialog box, highlight target class Function. Select possible target **Accept and Format Requests** and click the **Add** button. Click the **Done** button to close the dialog box.

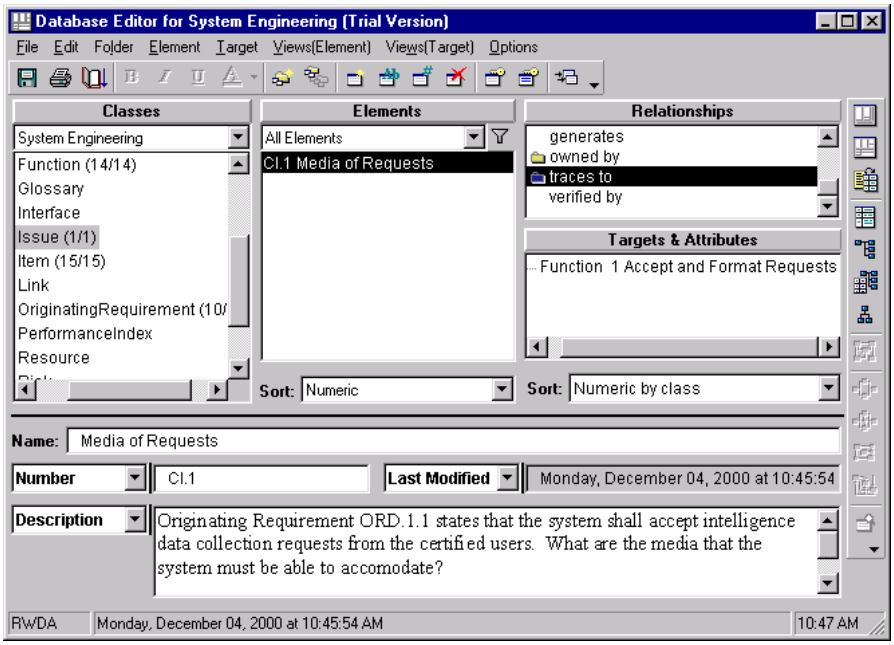


Go to the **Database Editor** again, from the **Views (Element)** pull-down menu, select **Hierarchy** and choose **Traceability** from Definitions. The results should be as shown.



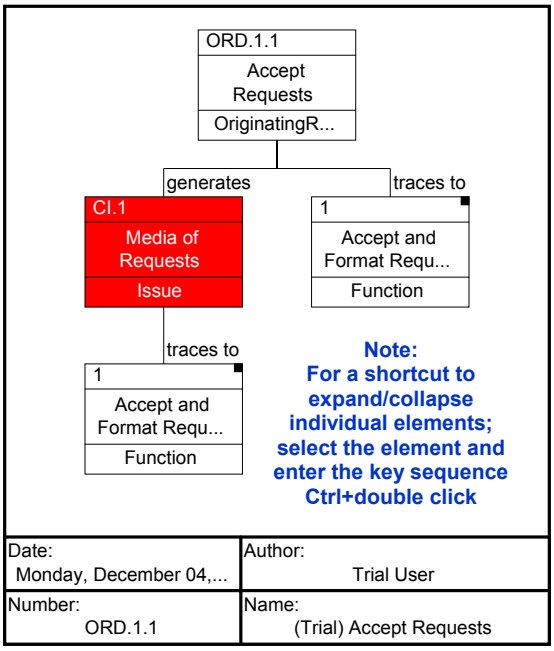
Extending Traceability

Continue by extending the traceability for the issue **Media of Requests**. In consecutive panes in the **Database Editor**, select class Issue, element **Media of Requests**, and relationships traces to. In the **Target** dialog box, highlight target class Function. Select possible target **Accept and Format Request** and click the **Add** button. Click the **Done** button to close dialog.



Now in the **Database Editor**, again select the class **Originating Requirement** and the element **Accept Request**, then from the **Views (Element)** pull-down menu, select **Hierarchy** and choose **Traceability** from the show list button.

You see the traceability of the **Originating Requirement** down to the Function level.



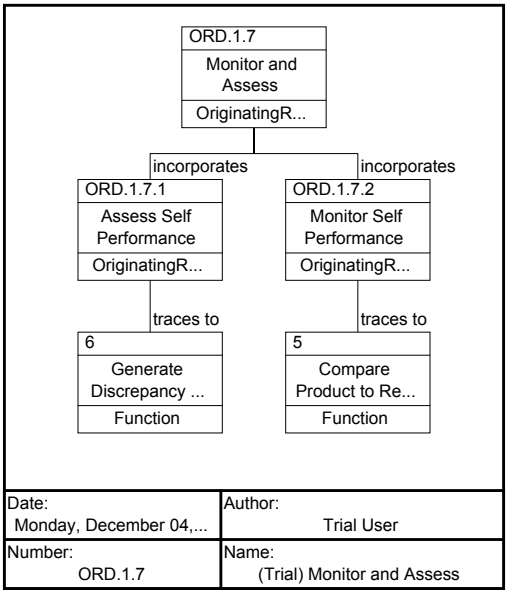
NOTE

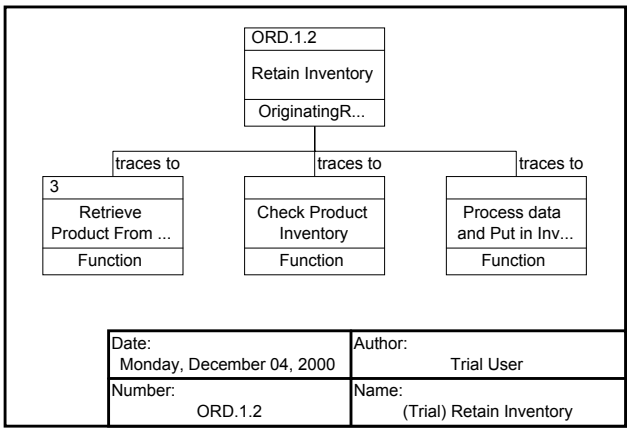
If any icons in a hierarchy diagram have a black dot in the upper-left corner, that icon can be further expanded. From the Settings menu, select the Diagram Options command and set the number of levels to a number greater than 3, which is the global default. A black dot in the upper-right hand corner of an icon means that icon is repeated somewhere else on that diagram. To see where else it appears, highlight it, and select the Highlight Matching Nodes command in the Diagram menu.

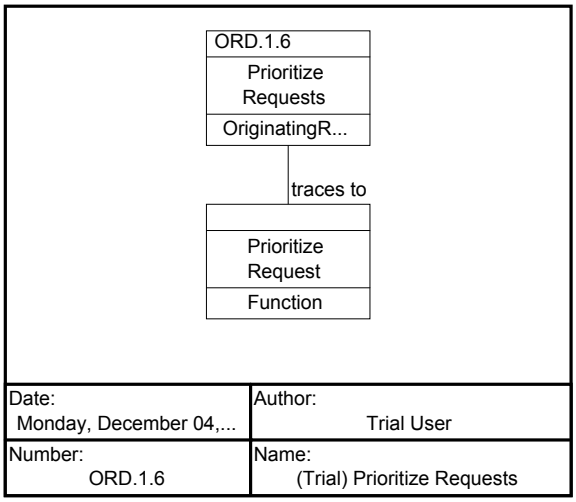
Extending Traceability (cont.)

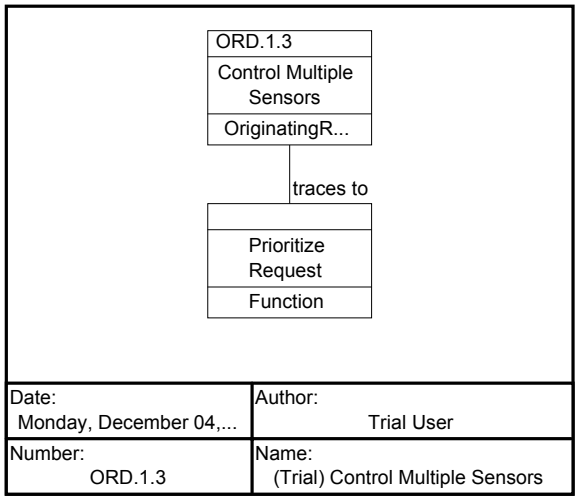
Use the same technique to extend traceability for the *Originating Requirements* **Retain Inventory**, **Control Multiple Sensors**, **Monitor and Assess** and **Prioritize Requests**. Use the diagrams below to guide you. While creating the **Monitor and Assess** diagram notice the function **Compare Product to Request** is traced from **Assess Self Performance** and the function **Generate Discrepancy and Recommendations** is traced from **Monitor Self Performance**.

The *Originating Requirements* **Maximum Staff** and **Provide Feedback** do not trace to any Functions in our model at this time.





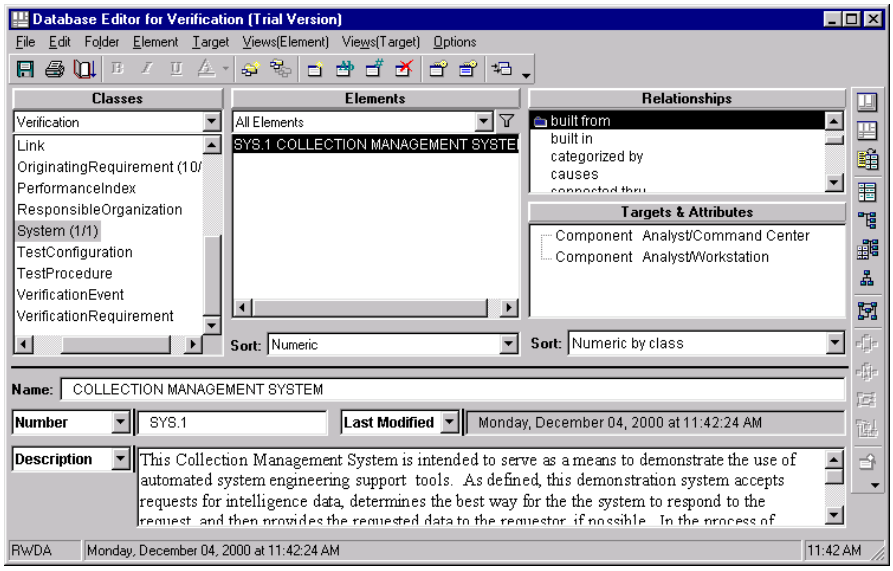




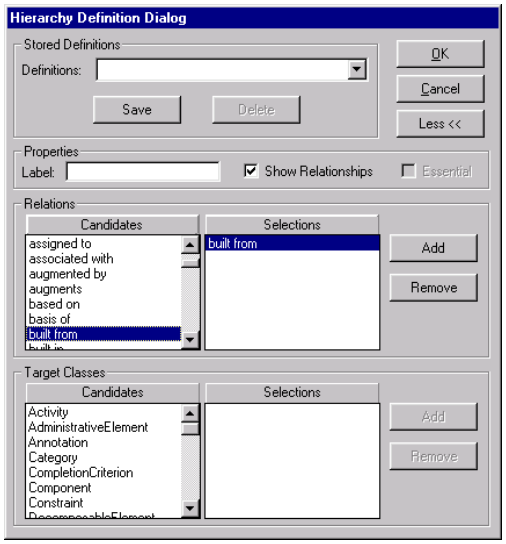
Extending the Component (Physical) Hierarchy

Physical Hierarchy:

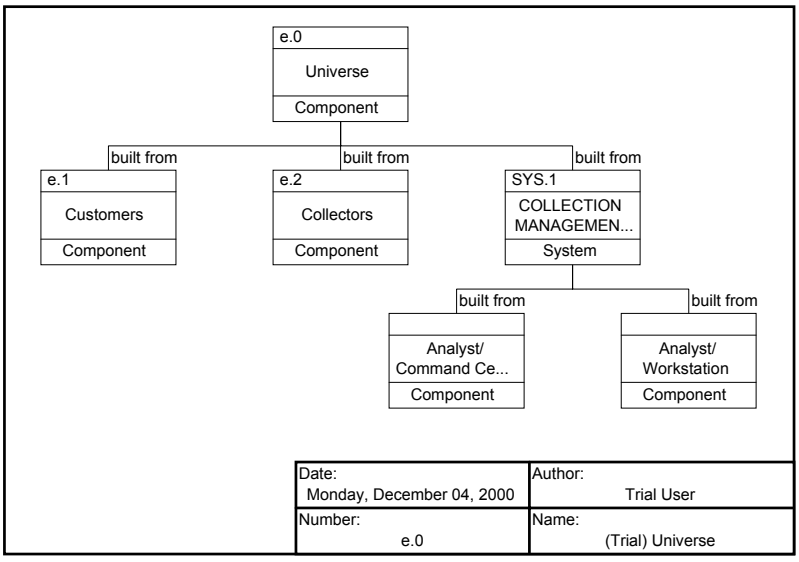
Let us now assume that our System is built from two components: **Analyst/Workstation** and **Analyst/Command Center**.



Use the **Database Editor** to add these targets using the relation built from. Now, open a **Physical Hierarchy** view of the **Universe** component. Do this by selecting class Component and element **Universe**.

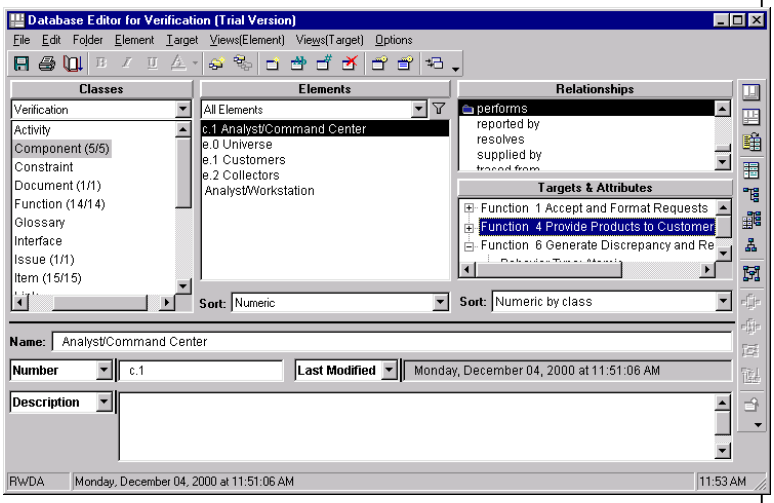


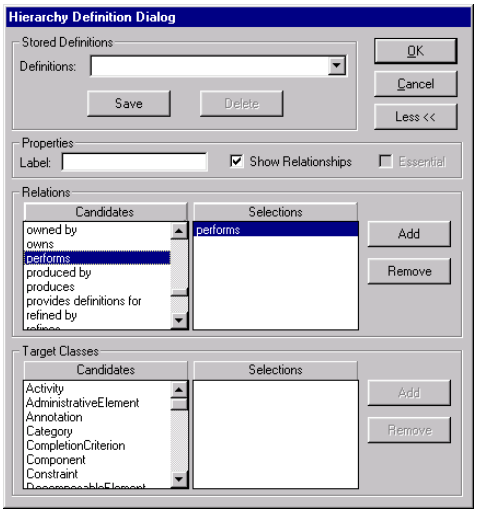
Go to the **View (Elements)** pull-down menu and choose **Hierarchy**. In the **Hierarchy Definition Dialog** box, choose **Physical** from the Definitions.

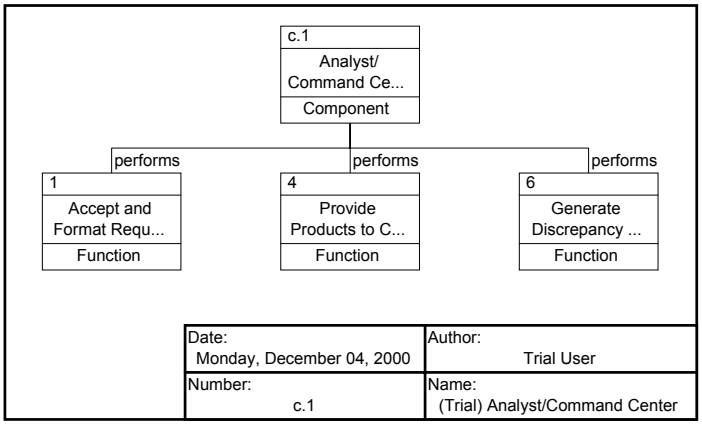


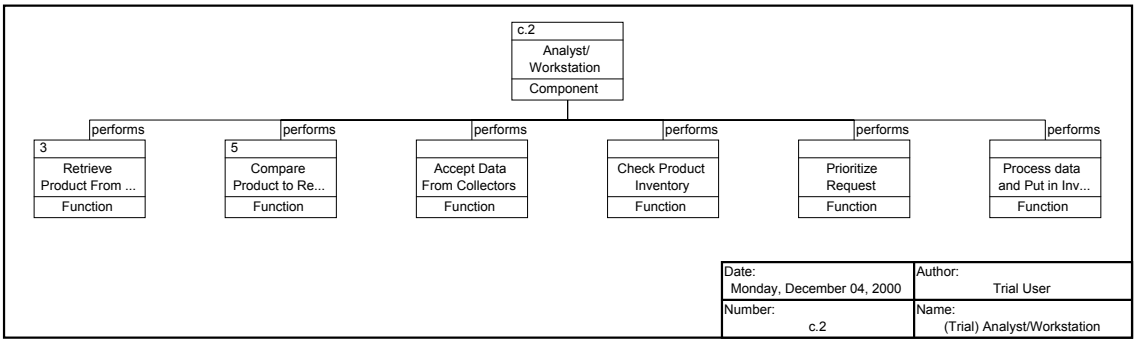
Allocating the Functions

Let us use the **Database Editor** to establish the relationship that these (leaf-level) components perform the (leaf-level) functions as shown in the two *custom* hierarchies below (See page 58). (performs is the inverse of the allocated to relation.)









Impact Analysis

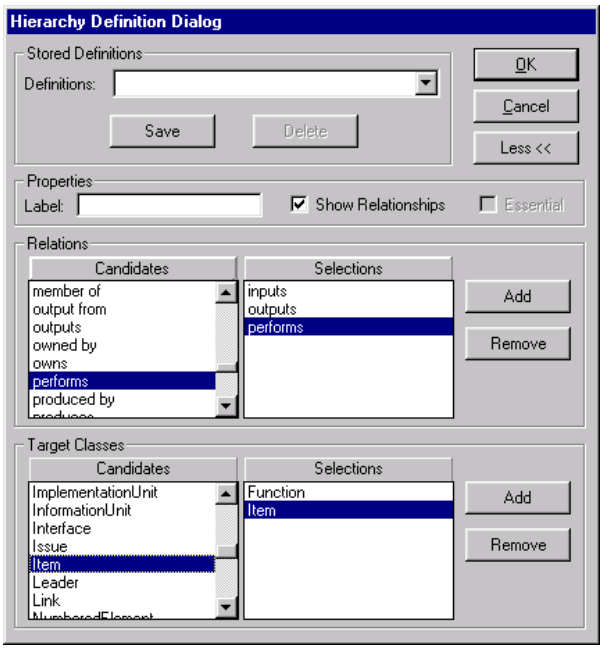
ERA diagram:

To show some of the power of our **ERA** design repository, let us postulate a typical system engineering example.

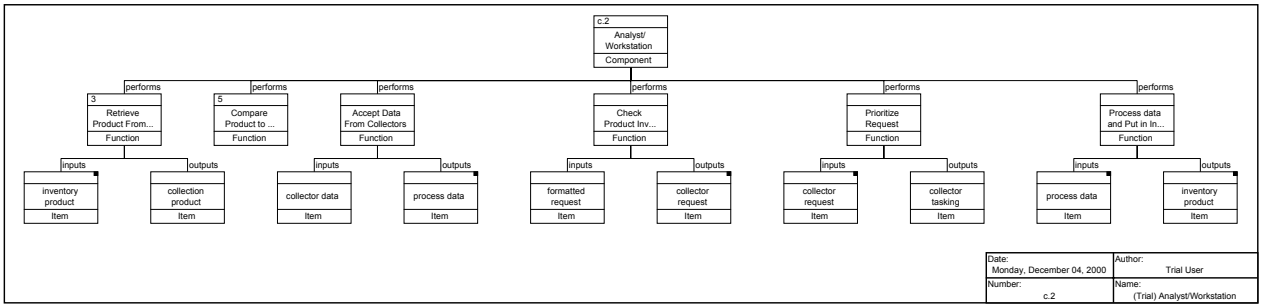
Assume that the customer wants to know the impact of exchanging with the workstation in the **Analyst/Workstation** component.

Select the component **Analyst/Workstation** and create a custom hierarchy with the performs, inputs, and outputs relations and the targets, Functions and Items.

Hint:
With the cursor highlighting any item in the candidate list, pressing the first letter of the desired candidate jumps to the first candidate beginning with the entered letter; example – pressing P within the candidate list jumps to performs

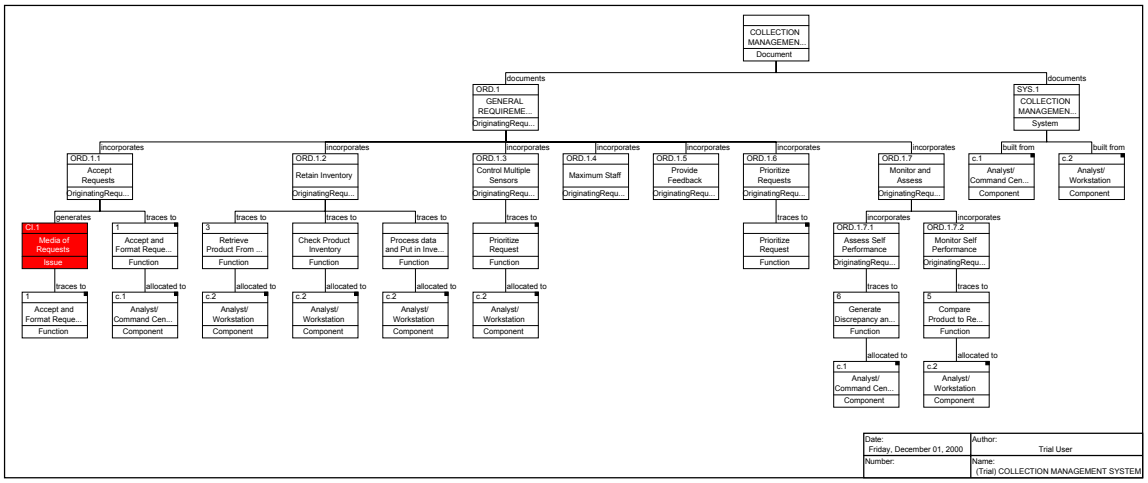


The diagram quickly shows which functions and which I/O (inputs and outputs) may be impacted.



CORE Provides Full Traceability from Source Document to Physical Architecture

The **Hierarchy** diagram below displays the document, **Collection Management System**. In the **Database Browser** select the class **Document** then select **ORD.1 Information Management System**. Click on the **Hierarchy** button and choose the stored definition **Traceability**. In the diagram, go to the **Settings Menu > Diagram Options** to set the level to seven and the scale to fit the window.



Using relationships for traceability makes it easy to detect unfulfilled requirements and unresolved issues. For example, Originating Requirements, **ORD.1.4** and **ORD.1.5** do not trace to functions on design elements.

Note
The Unallocated Leaf-level Requirements script may also be executed to produce an automated indication of which requirements do not trace to anything.

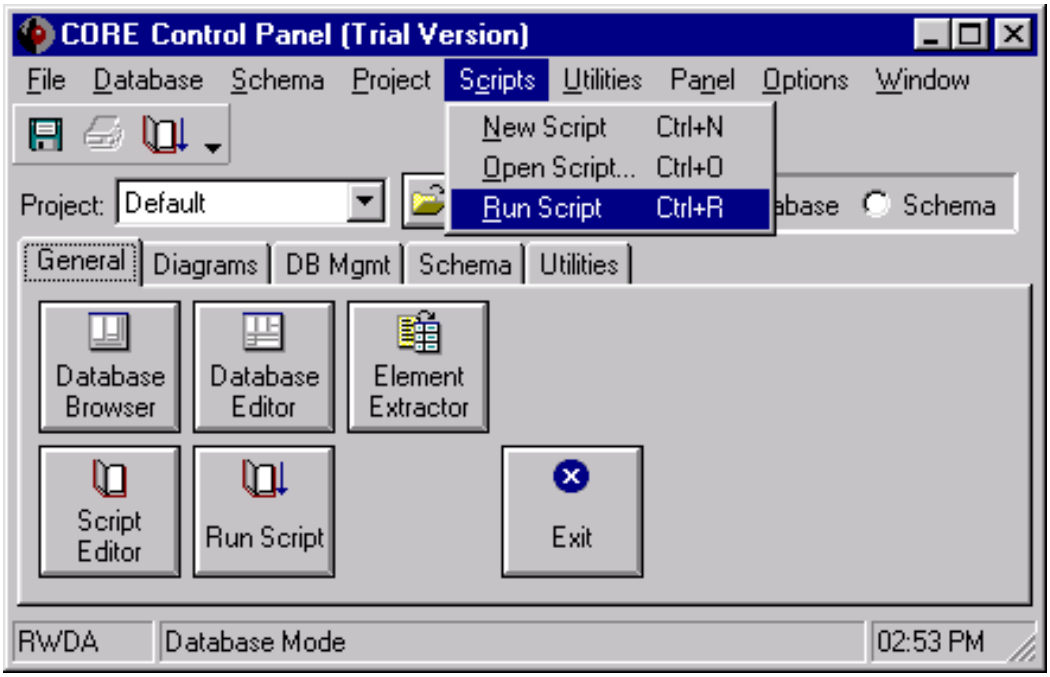
Generating a Report

An added benefit of **CORE** is its report writing capability. Reports in **CORE** can range from a simple database query (e.g. show me a list of all open issues) to complex, formatted reports (e.g. the System Description Document). **CORE** contains a number of built-in reports that can be executed directly in **CORE**. Also included in **CORE** is the **COREscript** language providing the capability to develop custom reports, queries, and interfaces to other tools. The built-in reports are written in the **COREscript** language and were created to satisfy some common database requests. Reports in **CORE** can be generated in any ASCII-based text file format. Most reports are generated as a file in Rich Text Format (RTF) (a standard publication file format) that can be imported into word processors such as Microsoft Word for preview, editing, and publication purposes.

The report generator enables you to extract system specification material from the **CORE** database repository and present it in virtually any desirable format. Reports allow you to view the database in different ways. The structure of a report is controlled by a report script that instructs the report generator where to go in the database to get data and how to format the information for each section of the report.

Generating a Report:

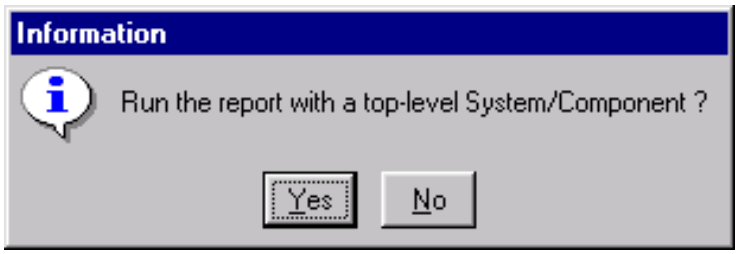
- From the **CORE Control Panel**, select **Scripts > Run Script**



Generating a Report (cont.)

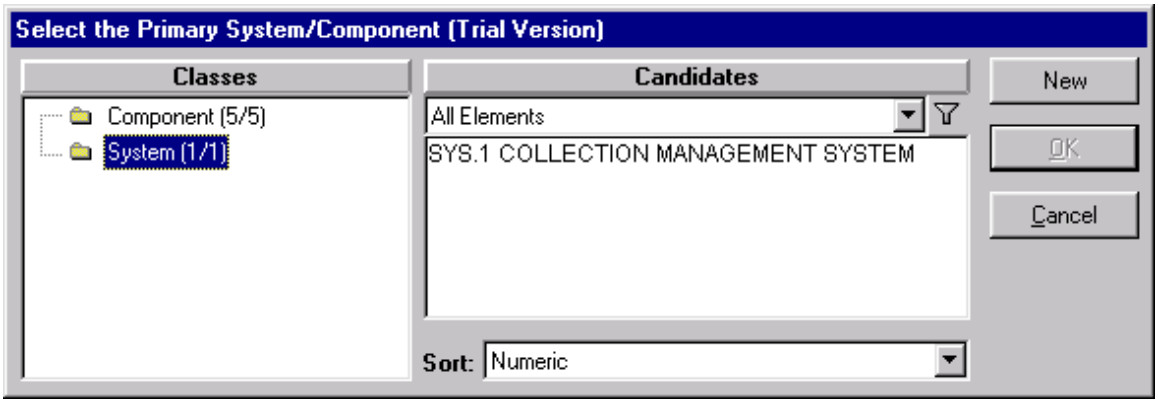
This opens a dialog prompt for selecting the desired report.

- Use the *pull-down arrow* to select **System Description Document**.
- Press **OK**.



A prompt will appear. Answer **YES** to running the report with a *top-level System/Component*. Next, you need to select the name of the **System** or **Component** in the repository for the report.

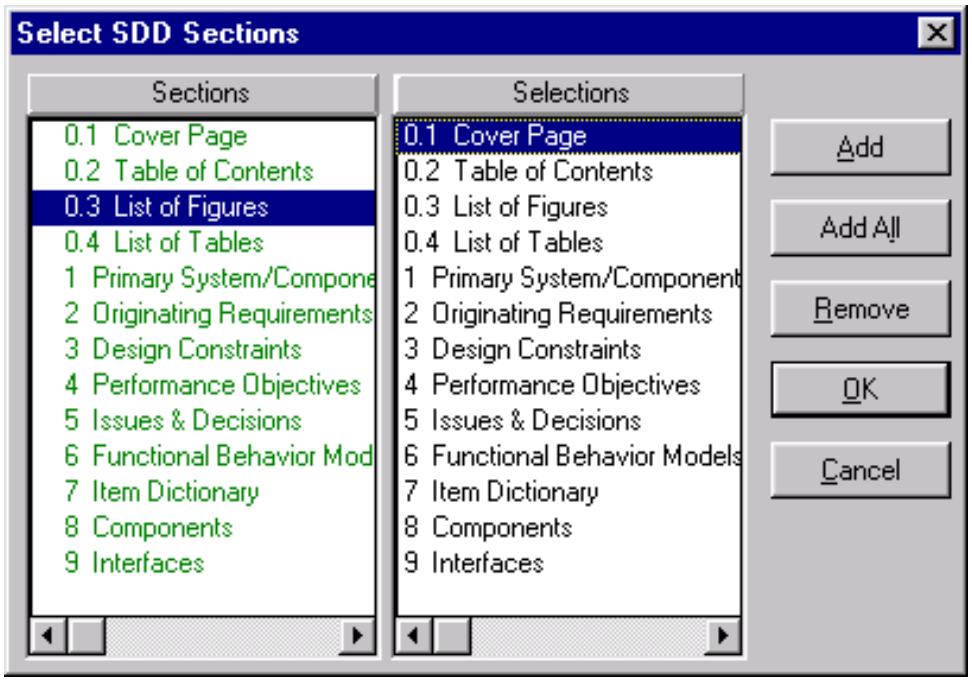
- Select **System** in the **Classes** pane
- Select **Collection Management System** in the **Candidates** pane
- Click **OK** button.



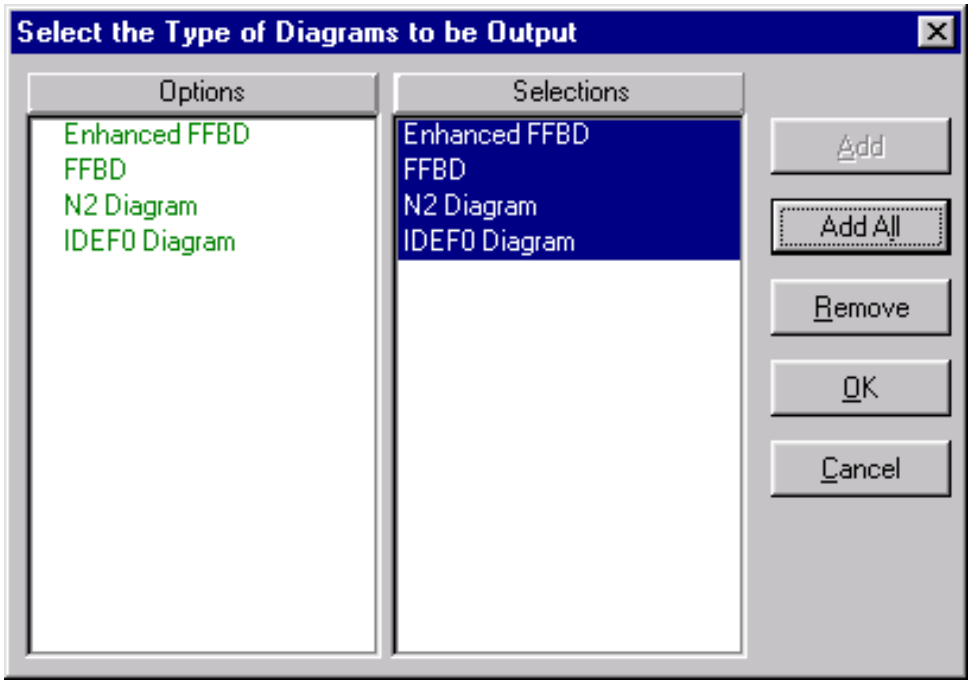
Generating a Report (cont.)

When creating the System Description Document (SDD), you can select the desired sections/parts of the SDD that you would like included. Select/ highlight the sections that you want in your report and click **Add**. If you want the (entire) default document, click the **Add All** button. We will include all the sections.

- Click **Add All**.



- Click **OK**.

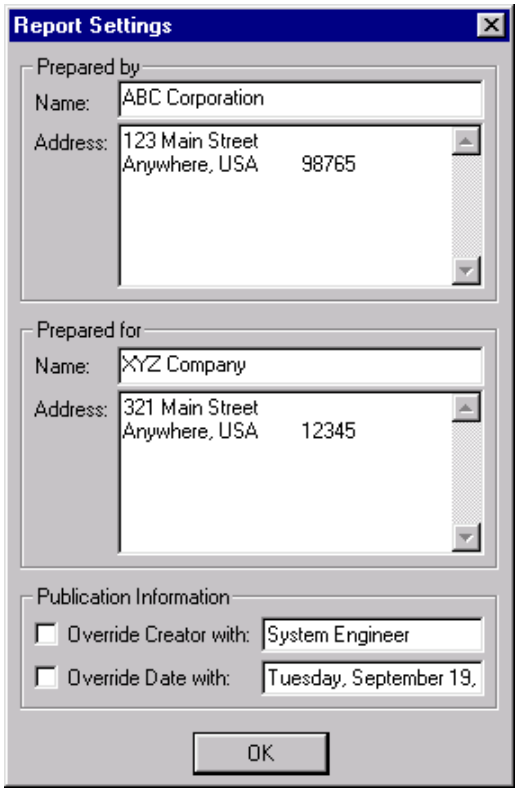


- Select **Add All** to include all diagram types.

Generating a Report (cont.)

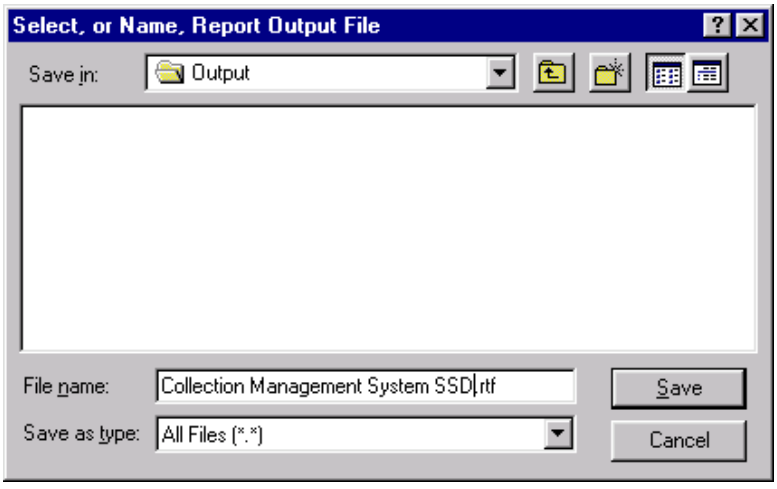
CORE displays a default Report Settings dialog box. These are the settings as defined in the **Report Preferences**. Update the settings from the **Preferences** menu it you'd like to have some names and addresses stored for later use.

- Edit the settings, if desired.
- Press **OK**.



Once you select which report you'd like, **CORE** walks you through a series of prompts to establish what you like to query.

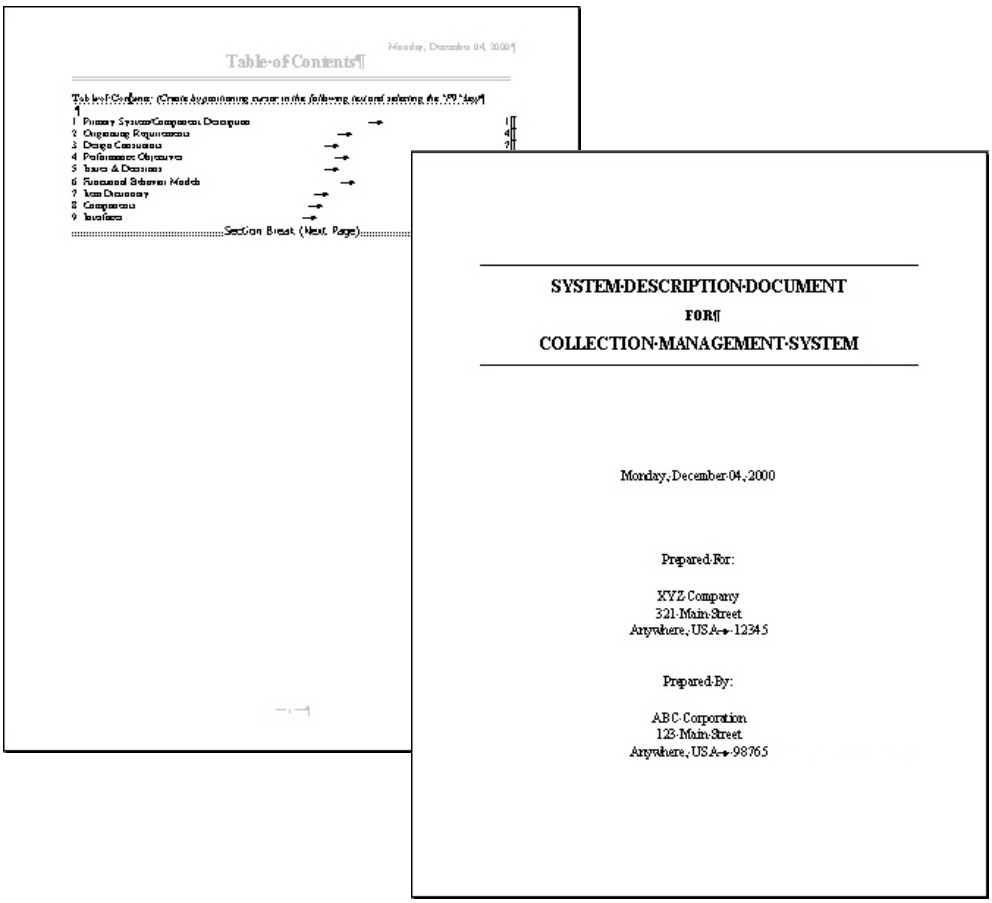
- From the prompt for a report name, type **Collection Management SSD**.
- Press **Save**.



CORE generates the report as an *RTF* (Rich Text Format) file. Import the RTF file into Microsoft Word (or your favorite publication tool or word processor) for previewing and printing.

Generating a Report (cont.)

The document will be completely formatted except for the **Table of Contents**, **List of Figures**, and **List of Tables**.



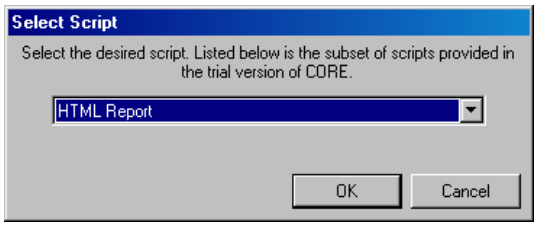
Highlight the identifier (**Body of Table of Contents**) and press the **F9** *function key* on your keyboard for each identifier. Word will automatically format and paginate for each table/list.

The System Description Report for this example is approximately 40 pages long.

Generating a Report (cont.)

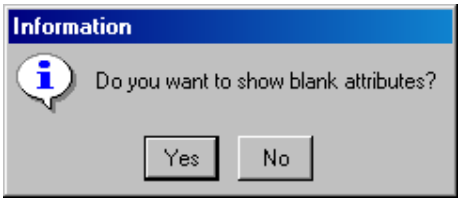
Another report available in **CORE** converts the System Engineering Database to HTML format and generates a Homepage for anyone with a Web browser to access.

- To run this report, select **Scripts > Run Script**
- From the **Select Script** pull-down select the **HTML Report**

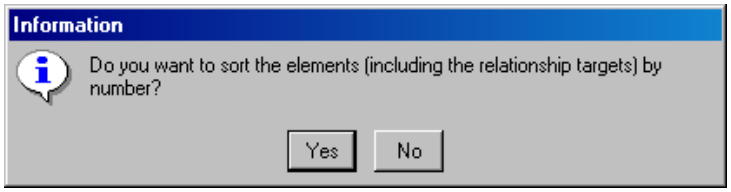


- When prompted to **Select a location for the Report Home Page** to be saved, click **Save** to accept the default filename (**0_homepage.htm**) in the default directory **(...\CORE 30 Trial\Output)**

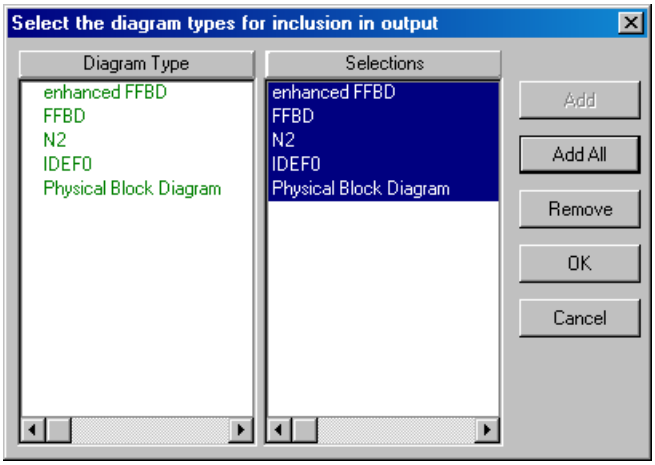
- When prompted to *show blank attributes*, click **No**



- When prompted to *sort the elements*, click **Yes**

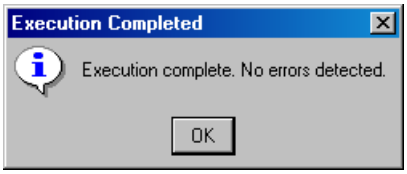


- When prompted to *select the diagram types*, click **Add All**
- Click **OK**



When the report generation is complete, the **Execution Completed** window appears

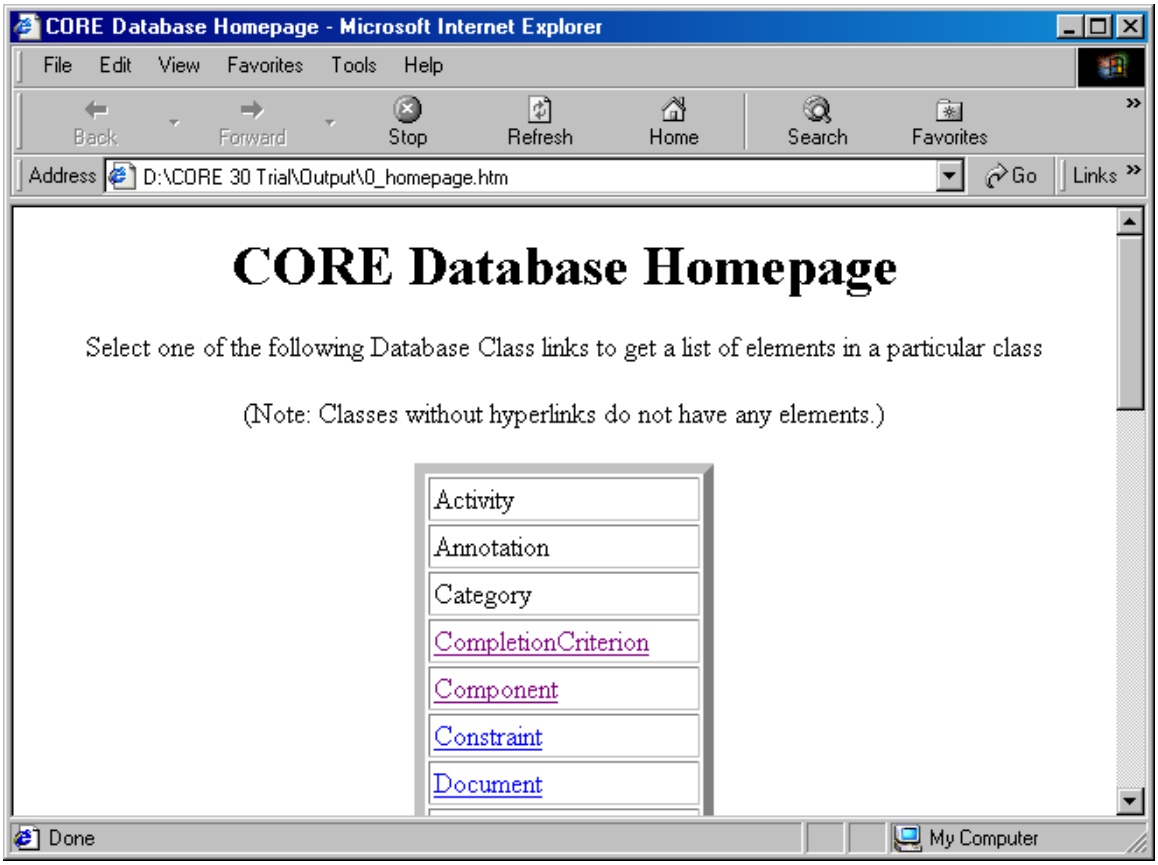
- Click **OK**



Generating a Report (cont.)

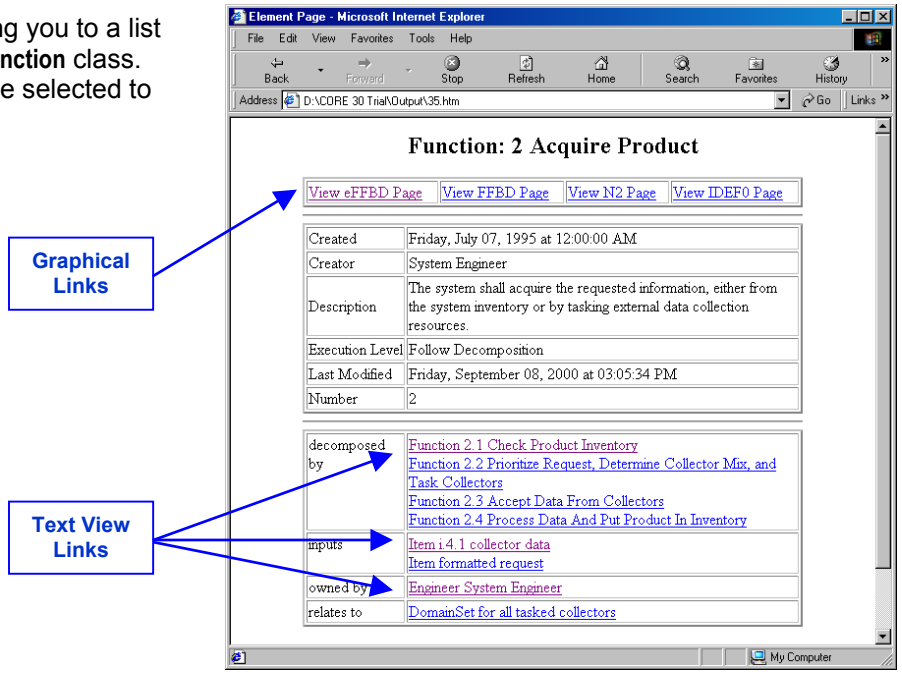
- Navigate to **\CORE 30 Trial\Output** and double-click the file **0_homepage.htm**

Your page should look similar to this image.



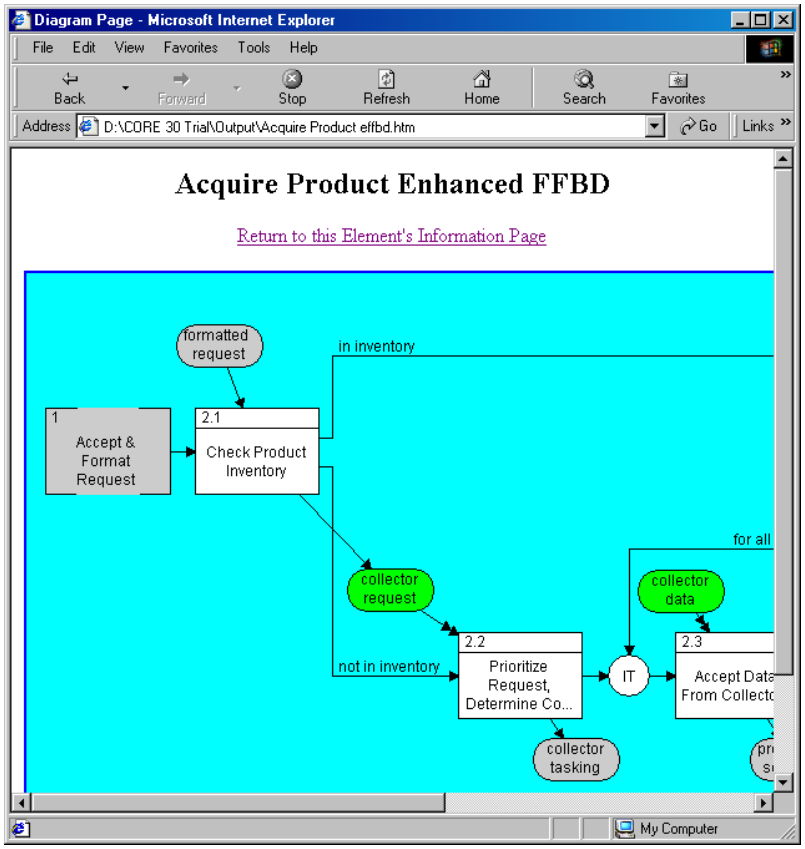
Generating a Report (cont.)

Clicking on **Function** bring you to a list of all *elements* in the **Function** class. Any function can then be selected to view its text view.



If the selected function has a diagram associated with it, that view is accessible by clicking on the related link at the top of the text view as shown above.

Once in the **Functional** view, you will notice that the graphical icons are also hyperlinked to their respective textual view.



CONGRATULATIONS

You have completed your first system design using **CORE**. Now, with this tour as a reference, experiment by designing your own system. Remember that **CORE** is far more powerful and flexible than we have shown in this simple example. Experiment with the other features and capabilities to get a better idea of what you can do. In fact, you will discover that with **CORE**, system engineering involves less perspiration (leaving more time for inspiration).

System engineering should be productive and fun. We believe that using CORE® is both.

Vitech Corporation

2070 Chain Bridge Road, Suite 320
Vienna, Virginia 22182-2536
(703) 883-2270 FAX: (703) 883-1860
Email: info@vtcorp.com
Support: support@vtcorp.com
WWW: http://www.vtcorp.com

CORE 3.0 Trial Limitations

The **Trial** version of **CORE** is identical to the commercial version with the following exceptions:

- The capability to save an image file, which stores the data with the program in order to eliminate the import/export cycle, has been disabled.
- The **Trial** will import/export **CORE** database files in binary format instead of *ASCII format*. This format does not permit merging of databases.
- The capability to export only the database changes has been disabled.
- Users are limited to a single project.
- Only a subset of the available reports has been provided. In addition, the capability to create and modify report scripts has been disabled.
- The capability to maintain a recovery log has been disabled.
- The bridge between **CORE** and **RDD-100** has been disabled.
- The **User/Group** Tool, which allows multiple users and groups to be created within the **CORE** environment, has been disabled.
- An artificial limit on the number of elements in each class has been added. The user is limited to a total of 275 elements. Each class is limited to 5 elements with the following exceptions:

50 OriginatingRequirements	35 Functions
75 Items	20 Components
15 PerformanceIndex	40 Constraints
20 Links	25 CompletionCriteria
10 DomainSets	10 Interface

3 in each Program Management (Unique) Class, which include:

Activity	Leader
Product	Program
Project	Work Unit (alias: Task)
Work Package	

An artificial limit on the number of schema extensions has been added. The user is limited to:

2 additional classes	4 additional relations	1 additional facility

However, the user is free to define as many attributes and relationships as desired.

This version of **CORE** is provided under special agreement for evaluation or academic use only. If you wish to use **CORE** for commercial or other purposes, please contact **Vitech Corporation** at (703) 883-2270 or via e-mail at info@vtcorp.com. **Vitech** has an official price list, which provides for indefinite and evaluation licenses for **CORE** as well as system engineering training classes and purchased technical services.

CORE 3.0 Product Family

Products

CORE

An integrated engineering approach for developing and conveying global process and product solutions on your desktop. **CORE** enforces consistency, interactively deriving and associating system behavior models with originating requirements and physical architectures.

CORE Enterprise

Server based application for concurrent multi-user access. It incorporates a client-server architecture and commercial object-oriented database to provide full **CORE** support to the Integrated Product Team environment.

CORE2net

CORE2net provides access to all information and models contained in a **CORE** database via Internet Browser. A separately licensed component of the CORE Enterprise Server, CORE2net turns your **CORE** Enterprise system into a Web server. CORE2net is a "CORE viewer" that does not require any special software to be installed on a user's workstation.

COREsim

A discrete event simulator option, which executes the process and physical model to provide an assessment of system performance and to verify the dynamic integrity of the conceptual design.