

**Systems  
Engineering  
Guided  
Tour**

---

Copyright © 1998-2015 Vitech Corporation. All rights reserved.

No part of this document may be reproduced in any form, including, but not limited to, photocopying, translating into another language, or storage in a data retrieval system, without prior written consent of Vitech Corporation.

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 252.277-7013.

**Vitech Corporation**  
2270 Kraft Drive, Suite 1600  
Blacksburg, Virginia 24060  
540.951.3322 FAX: 540.951.8222  
Customer Support: [support@vitechcorp.com](mailto:support@vitechcorp.com)  
[www.vitechcorp.com](http://www.vitechcorp.com)

 **CORE™** is a trademark of Vitech Corporation and refers to all products in the CORE software product family.

Other product names mentioned herein are used for identification purposes only, and may be trademarks of their respective companies.

Publication Date: June 2015

# Table of Contents

## Getting Started with CORE ..... 1

The Benefits of CORE ..... 1  
 Installing CORE ..... 2  
 Overview of the CORE Product Family ..... 2  
 Key Concepts ..... 3

## Examining CORE ..... 5

The Sample Problem: Geospatial Library ..... 5  
 Guided Tour Conventions ..... 5  
 Getting Started with CORE ..... 6  
 Importing CORE Data ..... 6  
 Project Explorer ..... 9  
 Element Browser ..... 10  
 Element Property Sheet ..... 11  
 Accessing a Graphical View ..... 12  
 Element Table ..... 13  
 Saving CORE Data ..... 14  
 Saving a CORE Repository ..... 15

## Starting a CORE Project ..... 17

The Sample Problem: Geospatial Library ..... 17  
 Summary of the Typical Top-down Process ..... 17  
 Capturing the Problem and Source Requirements .... 17  
 Parsing the Document Element ..... 19  
 Capturing the Elements into the Design Repository .. 19  
 Identifying Other Elements ..... 20  
 Defining a Relationship ..... 21  
 Establishing Traceability to the Source Document .... 22  
 Extracting the Child-Level Originating Requirements 24  
 Viewing a Requirements Diagram ..... 25  
 Viewing a Traceability Spider or Hierarchy Diagram . 26  
 Adding Elements in a Traceability Spider Diagram ... 27  
 Replacing Icons with Graphics ..... 28  
 Capturing Requirements Concerns ..... 29

## Defining the System and its Boundary ..... 31

Defining the System Environment ..... 31  
 Identifying System Level Interactions ..... 33

## Building the Behavior Model ..... 35

Creating Function Elements ..... 35  
 Building a Functional Model ..... 35  
 Inserting a Parallel Structure ..... 35  
 Adding Functions to an Activity Diagram ..... 36  
 Adding Inputs and Outputs ..... 36  
 Deriving the Behavior for Our System ..... 38  
 Adding Inputs and Outputs in an Activity Diagram .... 40  
 Adding to the Traceability ..... 43

## Completing the Physical Model ..... 45

Extending the Component (Physical) Hierarchy ..... 45  
 Allocating the Functions ..... 46  
 Completing the Physical Model ..... 48  
 Impact Analysis ..... 50

Ensuring Full Traceability from Source Document to  
 Physical Architecture ..... 50  
 Change Control ..... 51

## Generating Documentation ..... 53

Generating a System Description Document (SDD) . 53  
 Generating TeamView ..... 55  
 Congratulations! ..... 57  
 Standard Reports Provided with CORE 9 ..... 58

## Appendix—Using CORE University Edition ..... 61

Opening CORE 9 University Edition ..... 61  
 Importing a Data File ..... 62  
 Exporting (Saving) a Data File ..... 64  
 CORE 9 University Edition Features ..... 64



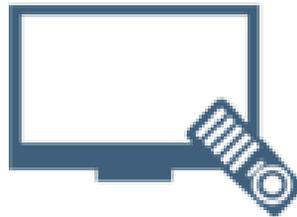
## CUSTOMER RESOURCE OPTIONS

Supporting users throughout their entire journey of learning MBSE is central to Vitech’s mission. For users looking for additional resources outside of this document, please refer to the links below. Alternatively, all links may be found at [www.vitechcorp.com/resources](http://www.vitechcorp.com/resources).



### [Webinars](#)

Webinar archive with over 40 hours of premium industry and tool-specific content.



### [Screencasts](#)

Short videos to guide users through installation and usage of Vitech software.



### [A Primer for Model-Based Systems Engineering](#)

Our free eBook and our most popular resource for new and experienced practitioners alike.



### [Help Files](#)

Searchable online access to Vitech software help files.



### [Technical Papers](#)

Library of technical and white papers for download, authored by Vitech systems engineers.



### [MySupport](#)

Knowledge Base, Exclusive Webinars and Screencasts, Chat Support, Documents, Download Archive, etc.

Our team has also created resources libraries customized for your experience level:

<a href="#">All Resources</a>	<a href="#">Advanced</a>
<a href="#">Beginner</a>	<a href="#">IT / Sys Admin</a>
<a href="#">Intermediate</a>	<a href="#">Student</a>

# 1

## Getting Started with CORE

In this section, you get an overview of CORE

- Benefits
- CORE Product Family
- Key Concepts

### The Benefits of CORE

Welcome to the CORE 9 Guided Tour. This guided tour is intended to familiarize you with basic features of CORE 9, the premier integrated systems engineering tool available today.

Developing complex systems requires more than what today's office software, requirements tools, and drawing packages can deliver. Product engineering and architecting demand a powerful support environment for life-cycle design. Whether designing a commercial product, an IT service, or a military system, satisfying diverse customers under schedule and budget constraints requires an integrated solution—a solution to synchronize requirements, analysis, and architecture; a solution to guarantee consistency and reduce risk; a solution to deliver technical and management insight into complex concerns and risks. That solution is CORE.

The CORE Product Suite is a fully integrated, flexible approach to collaborative product design specifically developed by systems engineers for systems engineers. Supported by an experienced staff of engineering professionals with real-world knowledge of the latest approaches and proven project experience, CORE puts project success first.

Unlike software tools focused on document-centric or diagram-centric approaches, CORE delivers a truly collaborative design-centric approach to product development. CORE provides comprehensive traceability from need definition through requirements and analysis to architecture and test. Built upon a proven approach and a central integrated design repository, CORE includes a comprehensive behavior modeling notation to better understand the dynamics of your design, integrated product simulation derived directly from your models, and on-demand automatic document and view generation. With numerous views tailored to the multitude of engineering and management tasks, CORE enables your team to focus on the creative aspects of engineering and system architecting.

Whether your project requires formal design specifications or informal web-based documentation, strict processes or agile design explorations, top-down approaches for a new system or middle-out/bottom-up reengineering of existing systems, CORE supports your needs.

CORE runs under the Microsoft® Windows® operating system and supports systems engineers during the phases of system definition, analysis, and design. The familiar interface makes CORE easy to learn and use. CORE allows efficient manipulation and representation of the system definition data.

This guided tour can be used in conjunction with the full version of CORE 9 as well as CORE 9 University Edition. Information regarding limitations found in the University Edition is provided in the Appendix found at the end of this guided tour. Some additional information about this guided tour:

- This guided tour provides a simple, structured walkthrough of a sample product engineering problem in order to introduce you to the basic concepts and capabilities of CORE. It is not intended to demonstrate the full power and flexibility of CORE.
- All versions include the sample database, the Geospatial Library, as it has been captured in the tool. The sample solution is presented in the data file `GeospatialLibrarySampleSolution.xml`<sup>1</sup> in the Samples directory. You can use this guided tour to recreate a portion of the solution yourself from scratch as we go along (starting on page 16).
- Help documentation and the CORE System Definition Guide that are installed with CORE 9 (accessible via the Help menu) provide in-depth information beyond that contained in this guided tour.
- Moving far beyond this introductory tour, Vitech offers several informational and training opportunities ranging from a one-day seminar for managers to a four-day course for practitioners and several options in between. Whether you are interested in a model-based systems engineering (MBSE) overview or a hands-on class in CORE, Vitech offers a course to meet your need. Please contact Vitech or visit our website ([www.vitechcorp.com](http://www.vitechcorp.com)) for more information on our training classes.

<sup>1</sup> The file extension for the University Edition data files is .a90

## Installing CORE

If you haven't yet installed the CORE software, please download the appropriate installation guide from our website at [www.vitechcorp.com](http://www.vitechcorp.com). The installation guides walk you through 1) installing the software, 2) obtaining a license and/or activation key, and 3) starting the software. If you are a previous CORE user, the installation guide also provides information about upgrading from previous releases of CORE.

If you have any problems or questions regarding installation, licensing, or training for your CORE 9 product(s), contact us:

- For installation or general customer support: [support@vitechcorp.com](mailto:support@vitechcorp.com)
- For licensing questions: [licensing@vitechcorp.com](mailto:licensing@vitechcorp.com)
- For information about our CORE University Program: [universityprogram@vitechcorp.com](mailto:universityprogram@vitechcorp.com)
- For information on systems engineering and CORE training, visit our website at [www.vitechcorp.com](http://www.vitechcorp.com) or contact your Account Executive. +1 540.951.3322 [info@vitechcorp.com](mailto:info@vitechcorp.com)

## Overview of the CORE Product Family

### CORE Essentials

CORE Essentials provides individuals, small teams, and distributed users a complete MBSE development solution ready for immediate use.

Enjoy a robust product that includes a rich requirements management capability, multiple modeling notations and integrated discrete-event simulation, comprehensive architecture analysis, verification and validation, and robust, on-demand documentation.

### CORE Spectrum

Incorporating all of the features found in CORE Essentials, CORE Spectrum adds comprehensive support for DoDAF 2.02 and SysML, providing a single vehicle that enables team-wide perspective and analysis and the industry-exclusive ability to deliver answers and insight in multiple formats, regardless of the input approach.

Whether working independently or as part of the collaborative enterprise team, CORE Spectrum provides the ultimate answer in capability and flexibility.

### CORE Server

Part of a large, complex, or data-rich effort? Add CORE Server as a remote central data repository and enable CORE Essentials and CORE Spectrum users the ability to operate independently offline or as part of the collaborative engineering team maintaining the system data in one concurrent database on the CORE Server. Easily maintained and time-tested with over a decade in use, CORE Server provides a secure, convenient gateway for the team to operate in unison, taking advantage of the team-wide consolidation of information.

As the engineering team centerpiece, CORE Server offers unparalleled ease of use, team-wide system-level insight, live access to the latest system changes, comprehensive analysis, and instant, thorough system-design documentation production.

### CORE2net Web Server

CORE2net extends your CORE Server environment to the web. As a separately licensed component of the CORE Server, the CORE2net web server allows you to query the current information contained in the CORE systems definition repository, enabling design changes and real-time sharing of the current design state. CORE2net enables inter- and intra-team collaboration at the enterprise level with the ability to set up appropriate access permissions: authorized team members simply log into the project website using a web browser such as Microsoft Internet Explorer®. Engineers no longer need to be co-located in order to participate in a design effort. Managers and other reviewers can access their data in familiar formats (tabular, graphical, and hyperlinked) from any location.

## Key Concepts

The underlying technology that drives CORE (including COREsim) is summarized below.

### Integrated System Design Repository

CORE’s integrated system design repository supports the many individuals who are adding, deleting, changing, and reviewing design information that results in the specification of a system. This centralization allows all team members to work from a common, controllable baseline. Additionally, this approach is key to providing consistency of the elements in the system design and assures that all design views (graphic and otherwise) are always synchronized and consistent.

### System Definition Language (SDL)

Our approach to attaining an explicit system specification is grounded in the use of the System Definition Language (SDL) provided with CORE. SDL is a formal, structured language which avoids the ambiguity inherent in using common English to define or specify a system. The precise meaning of each language concept is fixed and documented to enhance team communication and /assure unambiguous interpretation of specifications using this language. The data model repository is structured by the SDL which is being extended by the use, if needed. SDL is an Element-Relationship-Attribute (ERA) language augmented by graphical structures with semantic meaning. The SDL is based on the following primitive language concepts:

- o *Elements* (i.e., entities) correspond to nouns in English. Elements define objects and serve as the basic units in the system repository. CORE groups these elements into one of several classes (e.g., Component, Function, etc.) in the system repository.

- o *Relationships* are similar to verbs. To be precise, a relationship that defines a link between two elements corresponds to the mathematical definition of a binary relation. Relationships are not commutative, each relationship having a definite subject and object. However, for each relationship, there is a complementary relationship that defines the link from the object to the subject. For example, when you allocate a Function element to a Component element (using the *allocated to* relationship), CORE automatically creates the *performs* relationship linking the elements in the reverse direction.

- o *Attributes* further describe elements much like adjectives modify nouns. The attributes of an element serve to define critical properties of elements. For instance, attributes of a component would include the component number and component type.

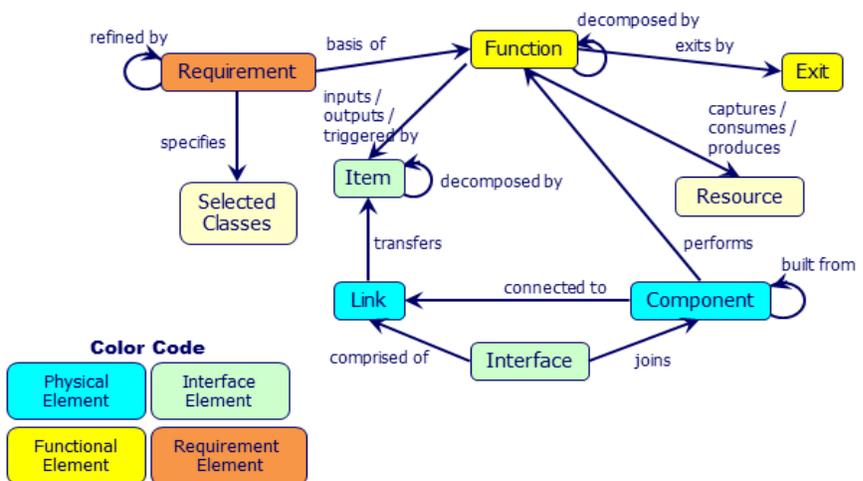
- o *Attributed-Relationships* (i.e., attributes on relationships) correspond to adverbs in English. The attributes of a relationship serve to define critical properties of the relationship. For instance, attributes of a *consumes* relationship would include the quantity being consumed.

- o *Structures* provide specification of semantically explicit system control constructs (Concurrency, Iteration, Loop, Multiple Exit, Replication, Selection, and Sequence). Using this explicit notation, the behavior of the system can be validated and shown to be executable using a discrete event simulator - COREsim. COREsim dynamically interprets a behavior representation so the simulation is always synchronized with the current model contained in the system design repository.

The data repository consists of elements that are modified by attributes and related to other elements. This structure corresponds to the object-oriented approach. Elements are represented as objects with the attributes stored as data within the objects. The relationships then define the interaction between objects.

In CORE, the SDL is referred to as a schema (or the project metadata.) The diagram below illustrates a subset of the basic schema, showing some of the primary systems engineering classes and relationships between them.

Primary Systems Engineering Elements



## Dynamic Graphical View Generators

CORE dynamically generates diagrams directly from the system design repository ensuring that they are consistent with current design details. A change made in any view changes the design information in the database repository and, conversely, a change made to the database repository is automatically reflected in the graphical views.

CORE delivers a mixture of traditional and SysML representations enabling you to satisfy the specific needs of your project. CORE provides a number of different graphical views (listed below) to meet the interests of both engineering and management personnel. This feature allows the system design to be viewed in as many perspectives and layers of abstraction as necessary to understand the model:

- Hierarchy and Spider Diagrams: graphically display several layers of relationships between elements on a single diagram such as functional, physical, and traceability hierarchy views.
- Package Diagrams: display arbitrary clustering of model elements to communicate groupings and interrelationships of interest
- Requirements Diagrams: show system requirements and their relationships to logical and physical components of the solution
- Use Case Diagrams: describe the functionality of a system in terms of how its users interact with the system to achieve their goals
- Functional Flow Block Diagrams (FFBD): show functional flow including control logic
- Activity Diagrams and Enhanced Functional Flow Block Diagrams (EFFBD): portray behavioral flow, control logic, and inputs/outputs/triggers
- Sequence Diagrams: represent the interactions between functions and their corresponding components
- Integration Definition for Function Modeling (IDEF0) Diagrams: show functions, inputs, outputs, controls, and mechanisms
- N2 (N-squared) Diagrams: display functions/components and their internal and external interactions in a matrix format

- Block Definition Diagrams: show composition and classification of the physical architecture
- Physical Block Diagrams, Interface Block Diagrams, and Internal Block Diagrams: show composition and connectivity (both physical and logical) of the physical architecture

## Automatic Document Generation

The CORE report generator enables you to extract information from the CORE system design repository and present it in virtually any desired format. Reports allow you to view the system design information in different ways. Reports in CORE can range from a simple query (e.g., a list of all open concerns) to complex, formal documents (e.g., a System/Segment Specification). Reports and analyses for engineering or management support are generated through the use of more than 50 standard utilities, queries, and report templates<sup>2</sup> provided with CORE. Reports in CORE can be generated in any ASCII-based text file format. Most reports are generated using Rich Text Format (RTF) (a standard publication file format) that can be imported into word processors such as Microsoft Word® for previewing, editing, and printing.

The structure of a report is controlled by a report script that instructs the report generator how to query the system design database repository to gather data and format the information for each portion of the report. Users can develop custom reports, queries, and interfaces to other tools as well as customize the standard scripts provided with CORE. Scripts are written using the COREscript language. Full documentation of COREscript can be accessed via the CORE Help menu. Also, Vitech offers a two-day class that focuses on exploiting the power of the COREscript language.

<sup>2</sup> A list of the full set of reports provided with CORE is shown on page 59. A limited set of reports is provided with CORE 9 University Edition. Refer to the Appendix for details.

# 2 Examining CORE

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

- Launch CORE
- Import a Data File into CORE
- Look at Basic CORE Windows
- Export (Save) a Data File from CORE

## The Sample Problem: Geospatial Library

In this section, we will use a sample database for the Geospatial Library to provide an overview of CORE and its features. The context diagram below provides a high-level view of the system we will use throughout this guided tour. You may find it helpful to refer back to this diagram as you build the system structure while working through the guided tour.

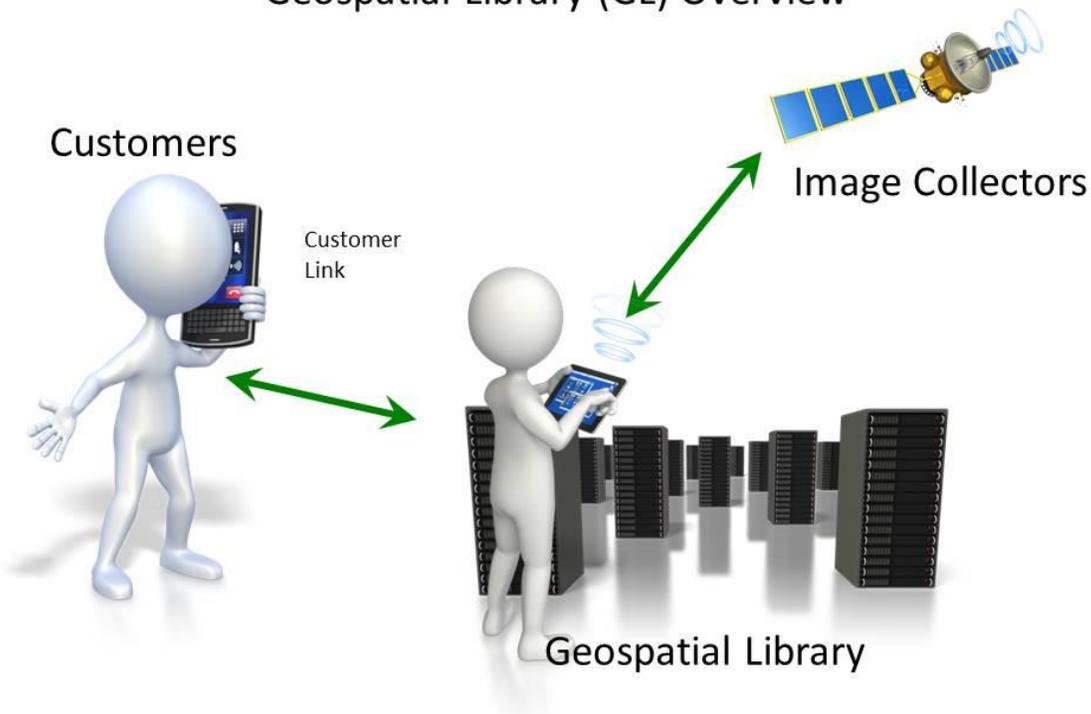
This Geospatial Library is intended to serve as a means to demonstrate the use of automated systems engineering support tools. As defined, this demonstration system accepts requests for imagery information, determines the best way for the system to respond to the request, and then provides the requested information to the requestor. In the process of acquiring the requested information, the system may generate tasking orders for a set of imagery data collectors.

## Guided Tour Conventions

The following special styling is used throughout the guided tour to help you navigate.

Font Style	Description
<b>Function</b>	Classes
<b>Number</b>	Attributes
<u>allocated to</u>	Relationships
<b>Contact Emergency Response</b>	User selections and inputs
<i>File &gt; Import...</i>	CORE commands, buttons, icons or tabs

### Geospatial Library (GL) Overview



## Getting Started with CORE

This section will show how to get started with CORE including importing a sample file, an overview of the basic menus and features, and saving your work.

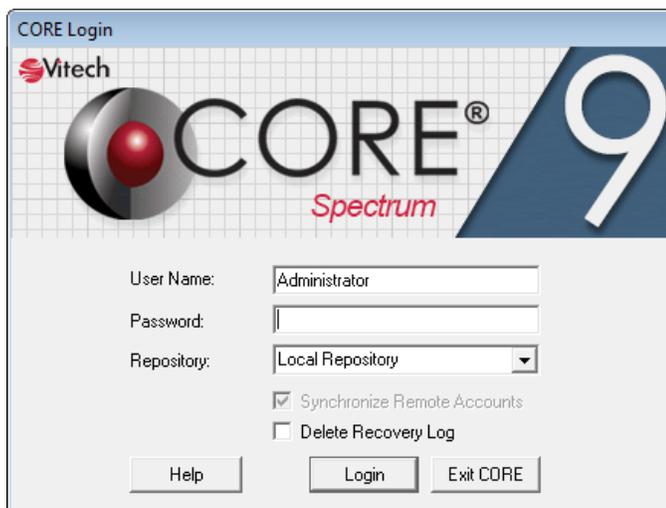
CORE University Edition users should go to the Appendix on page 61 to learn how to get started with CORE.

Once you have installed CORE, launch the CORE 9 application.

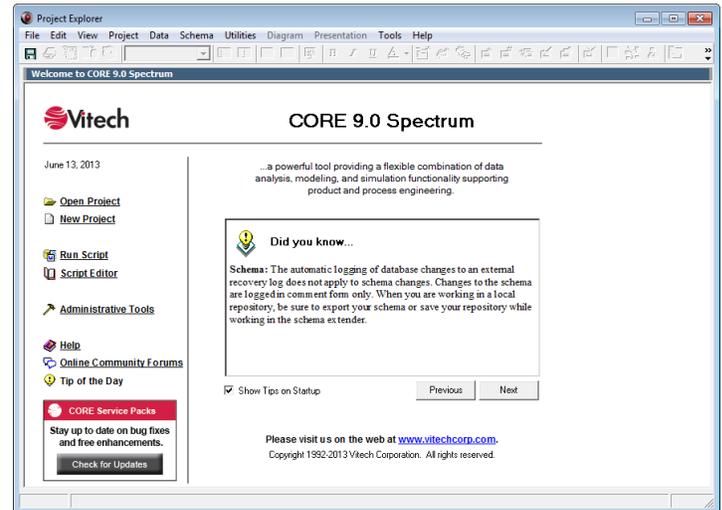
- Click the Windows **START** button.
- Select **All Programs**, open the **CORE 9** folder, and click **Empty Repository**. While CORE is being loaded, you will see the following screen.



- When prompted for a password, enter the default password **admin** and click **Login** or press the **Enter** button.



When you have successfully logged into CORE, the Project Explorer window will open with the welcome display (as shown below).



When you open a CORE repository, you are logged into a CORE database structure made up of one or more projects. A CORE repository file can contain multiple projects with each project containing its own set of data (referred to as the system design repository). You can select an existing project or create a new project either directly by importing an existing project or by selecting **New Project** and entering a project name.

### NOTE:

Each open project in CORE will have one or more corresponding Project Explorers open. When you close the last Project Explorer, you will be prompted with a message that you are exiting CORE.

## Importing CORE Data

To introduce you to CORE, we want to import a project so we can see the CORE system design repository populated with data. We will use the Geospatial Library sample project which has already been created in CORE and then exported to the \CORE 9\Data\Samples directory. In general, this import/export capability of CORE allows you to transfer a CORE project from one computer to another or make a backup copy of the data.

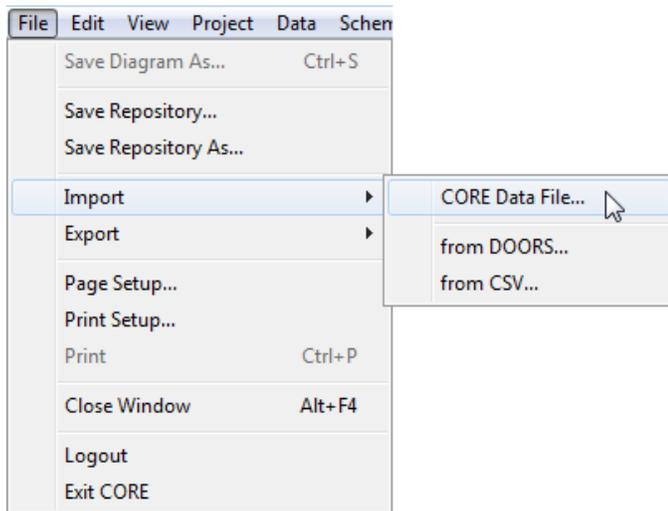
## 2: Examining CORE

In the next section, Starting a CORE Project, you will learn to use CORE by going through the steps to build this same sample problem yourself.

Users of the CORE University Edition should go to the Appendix on page 62 to learn how to import CORE data in the University Edition of CORE.

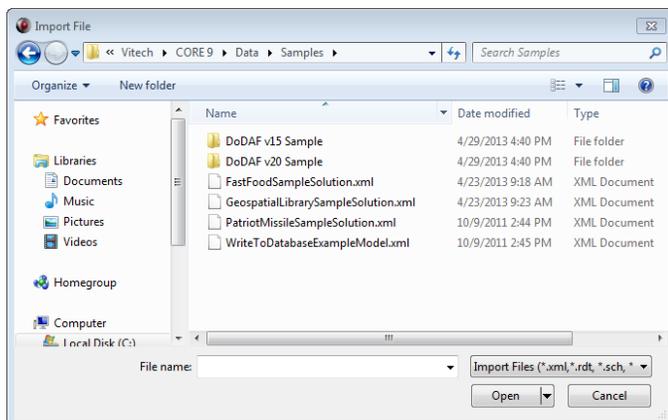
To import a CORE Data File:

- From CORE's Project Explorer drop-down menus, select **File > Import > CORE Data File**.



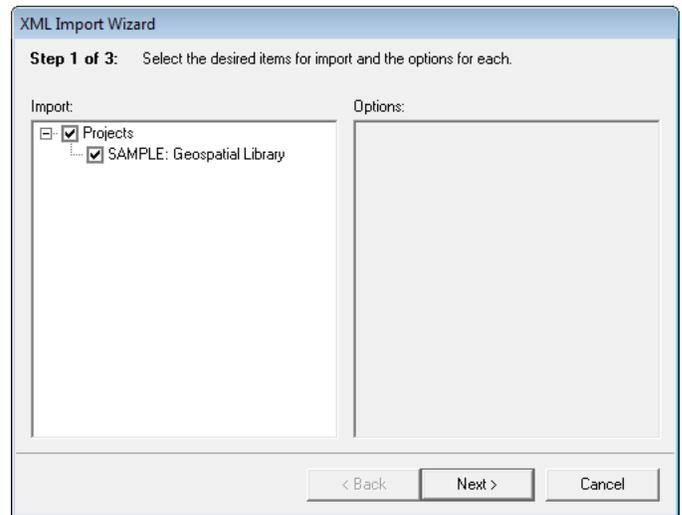
This opens the Import File dialog (see below).

- Navigate to the **CORE 9\Data\Samples** directory and select the file named **GeospatialLibrarySampleSolution.xml**.
- Click **Open**.

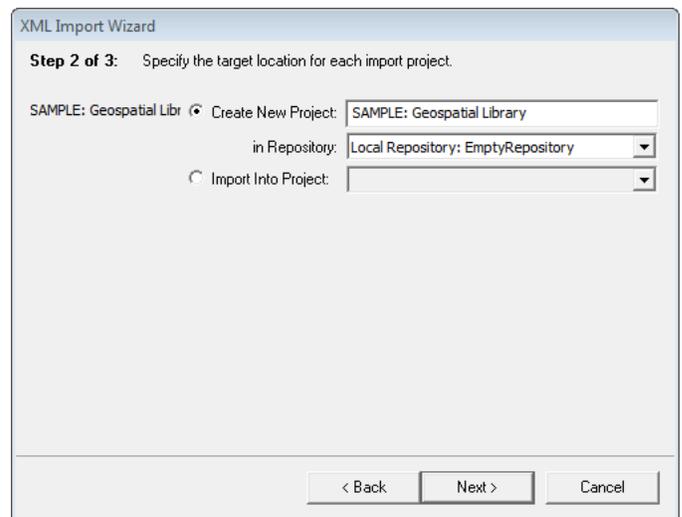


Next, the XML Import Wizard will appear allowing you to control the import.

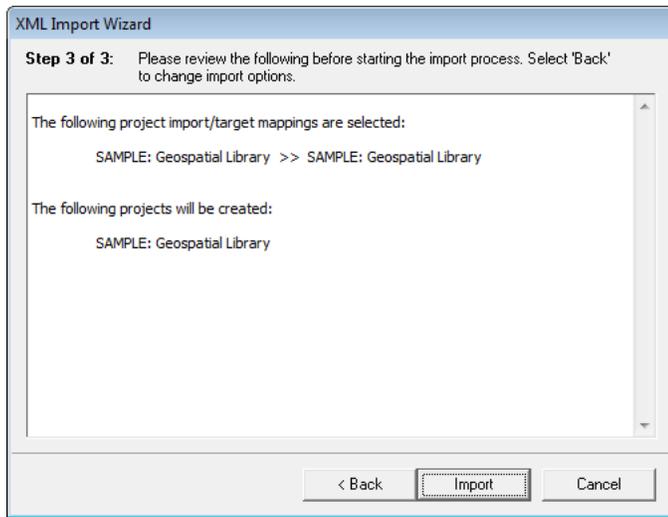
- In Step 1, click **Next** since our XML file contains only one project.



- In Step 2, click **Next** since we want to create a new project rather than importing into an existing project.



- In Step 3, click **Import** to begin the import.



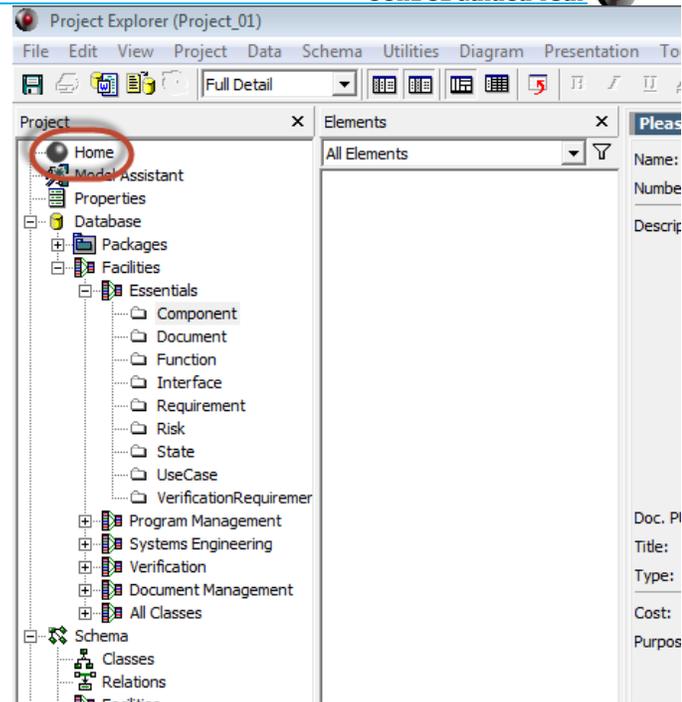
A status window will show you the progress of building the new project and importing the project data.

- Click **OK** when the status pane closes and the notification box opens as shown below..



We will now open the newly created project.

- Click Home in the Project Pane



- Select **Open Project** in the Project Explorer and double-click **Geospatial Library [Local Repository: EmptyRepository]** from the list of projects. Note that the title of our Project Explorer window now includes the name of the project.

# Project Explorer

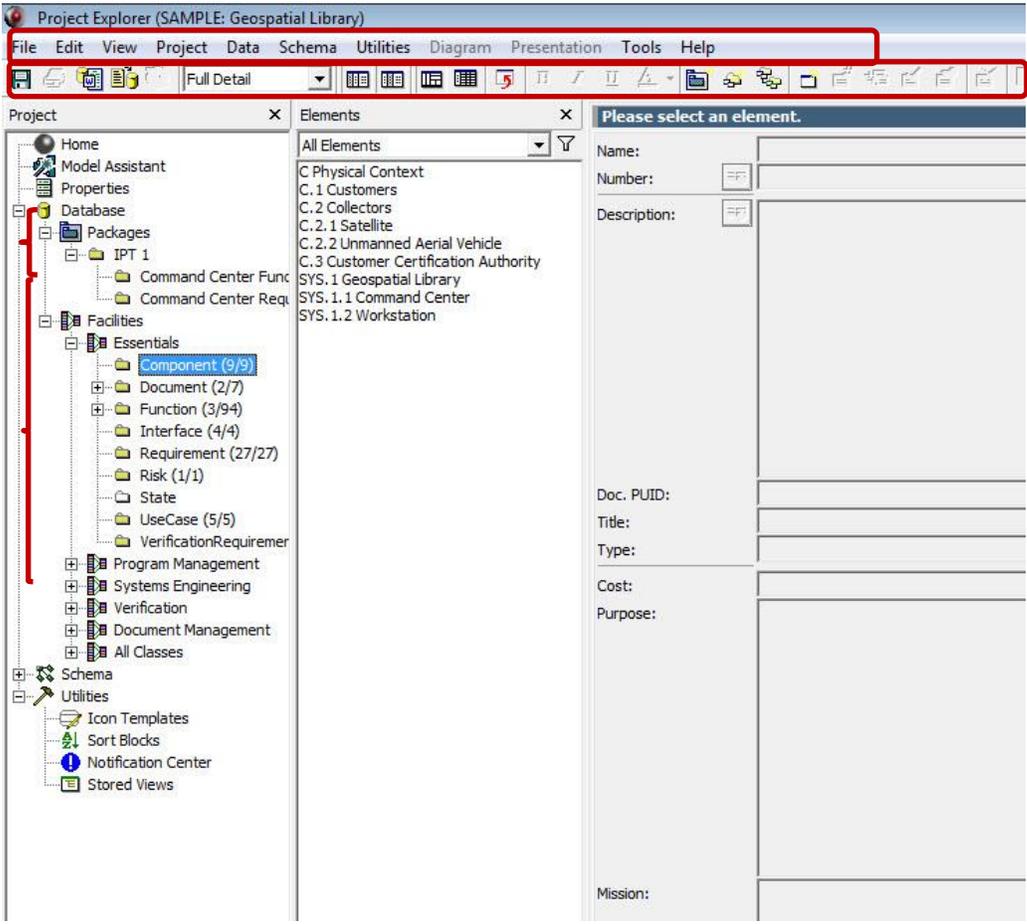
The Project pane is now visible on the left listing the classes and folders in the system design repository. The Project Explorer provides quick access to information contained in the system design repository for a particular project. Context-sensitive toolbars provide icons to access frequently-used commands.

**NOTE**  
 When starting a new project, users can select the schema that best meets their project needs—the basic systems engineering schema or the Department of Defense Architecture Framework (DoDAF) schema.

Menu  
Toolbar

Packages

Facilities/  
Folders



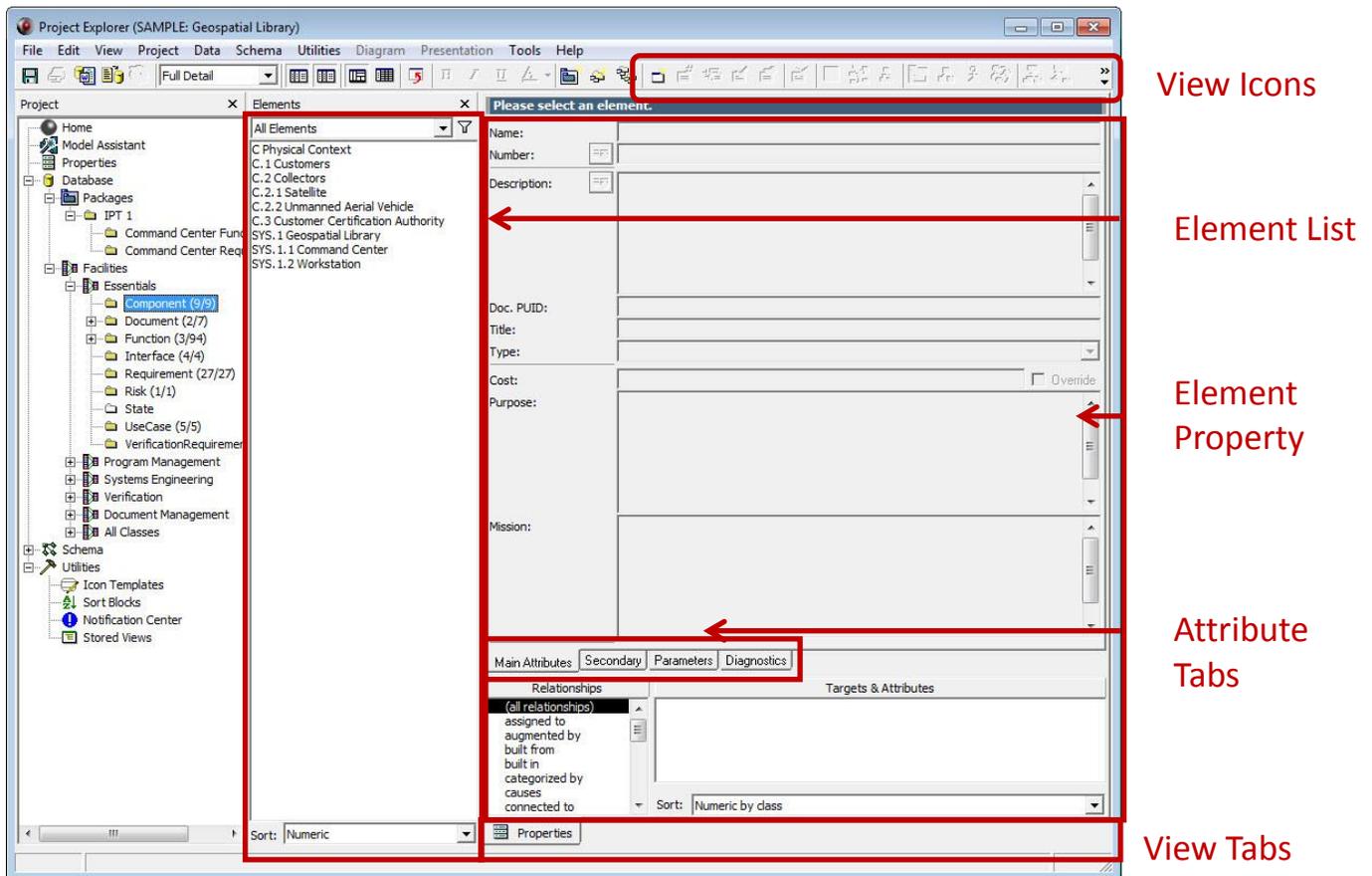
# Element Browser

Selecting one of the classes/folders in the Project pane displays all of that folder's elements in the Elements pane within the Element Browser. Once a Class is selected in the Project Pane, the Element List is visible in the Element Browser. In the Element Browser you can view the structure of the data, create new elements, and update elements. A tan folder preceding the class name in the Project pane indicates that at least one element (instance) of that class has been defined. The numbers in parentheses indicate how many elements have been defined for that folder and how many total elements in that folder and all of the subfolders.

- Select **Component** in the Project pane.
- Next select the **Component** named **Geospatial Library** in the Elements pane.

The Property Sheet for the **Component** will be displayed. Notice that the toolbar now displays an icon for each view that can be displayed for a **Component** and that the view tabs display the same icons.

The Model Assistant is always on by default and is causing CORE to display the view tabs for logical views because this element has a root function. To learn more about the Model Assistant, reference the Model Assistant screencast found on our website at [www.vitechcorp.com](http://www.vitechcorp.com).



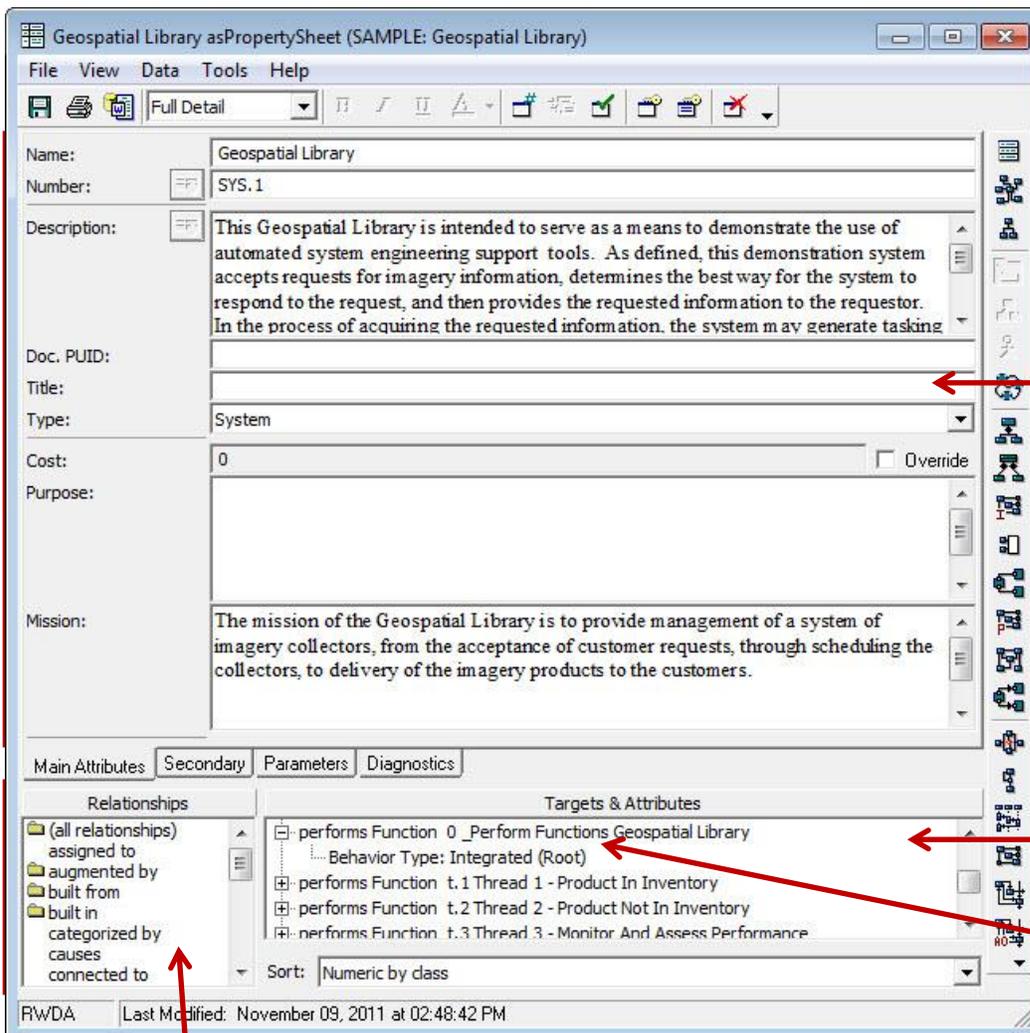
## Element Property Sheet

A Property Sheet provides the complete definition of a given element in the system design repository by displaying all the attribute values and relationships. You can use the Property Sheet view corresponding to any element to view, add, or make changes to the attributes and relationships of the displayed element. The list of attributes and relationships differs depending on the class of the element displayed. Here we are looking at an element of the **Component** class, so only those attributes and relationships that pertain to **Components** are displayed.

- Open the Property Sheet as a separate window by double-clicking on the element name, in this case: **Geospatial Library** in the Elements pane or, with the element selected, click on the *Property Sheet* icon  in the toolbar.

The attributes and their values are displayed in the upper portion of the sheet. 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.

- Scroll through the attributes. The most frequently used attributes are shown on the *Main Attributes* tab. Additional detail can be specified on the *Secondary* tab.
- Scroll through the list of possible relationships. A folder in front of a relationship indicates that a target has been established for the relationship.
- Click on performs. The toggle icon in front of the target indicates that the relationship has an attribute. Use the toggle to collapse/expand the list.
- Close the Property Sheet.



Attributes

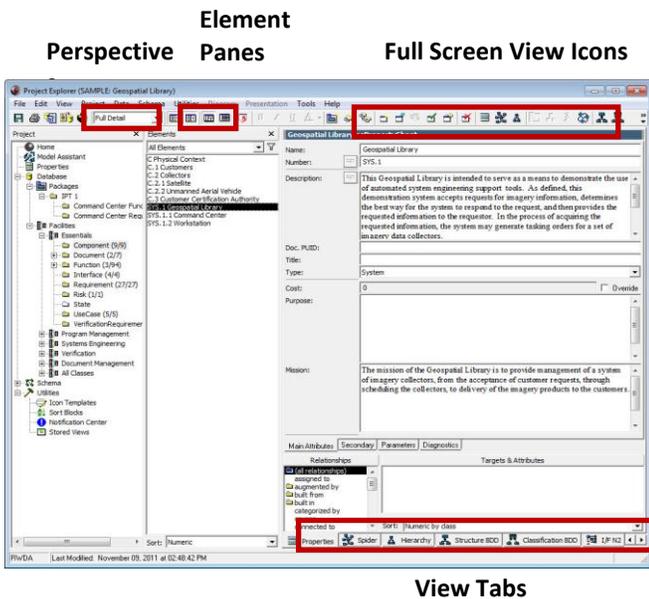
Relationship Target Elements

Relationship Attribute

List of possible relationships

## Accessing a Graphical View

In the Element Browser, you can directly access the many views of your element via either view tabs, view icons, or the *Open Element* submenu of the Elements pane pop-up menu. Using the bottom view tabs opens the view within the Element Browser. To create more room for a tabbed view, you can hide the Project pane and/or Element pane by using the *Hide/Show Project Pane* and/or *Element Pane* icons on the tool bar. Clicking a view icon on the toolbar opens the view in a separate window.



screen, click the *Structure BDD* tab

 **Structure BDD** to display the Structure BDD of the element (or click

the  icon to open a Structure BDD in a separate window).

**NOTE**

Throughout the guided tour, we will often open a separate view in order to enhance the readability of the view contents in this document. Where this occurs, you can display the view in the *Element Browser* by clicking the corresponding view tab (in the case of Hierarchy Diagrams, you must also select the type of hierarchy from the drop-down menu). Once the view is displayed, the step-by-step instructions to modify the view will still apply.

Any diagram view will have Diagram Elements and Diagram Palette panes that can be shown or hidden

using their respective icons .

The Diagram Elements pane (at the bottom of the diagram) is available to show selected attribute and relationship information about the elements in the diagram. The information is grouped by class and can be displayed in tabular form by using the toggle icon in front of the class name. As with information in all views, the attributes and relationships are directly editable.

The Diagram Palette (on the right-hand side of the diagram) is available to manage the diagram. By selecting a **Construct** and dragging it to the desired position on the diagram, it can be added to the view. By selecting the **Key Entities** tab, classes and elements that can be added to the view are displayed. Using drag-drop will add the selected element. By selecting the **All Entities** tab, all classes and elements are available and can be related to and elements in the view using drag-drop. Any action using the **Diagram Palette** will also update the underlying CORE database.

- If you opened a separate window to view the Structure BDD, please close it.

**NOTE**

Double-clicking on an element in any diagram view opens that element's Property Sheet in a separate window.

**NOTE:**  
The steps below require a CORE Spectrum license. If you have a CORE Essentials license, then you can view any diagram on a component element for this step. For future occurrences, you'll see this view icon accompanied by information for CORE Essentials users.

Let's look at an example. A Structure Block Definition Diagram (Structure BDD) represents the composition structure of systems, components, items, conceptual entities, and logical abstractions. Part of the physical architecture representation set, this is similar to a traditional physical hierarchy with select semantic and representational differences. Most notably, the Structure BDD can optionally indicate the role that the part plays in its parent.

- Select **Geospatial Library** from the list of elements in the **Component** class.
- In list of view tabs along the bottom of the

## Element Table

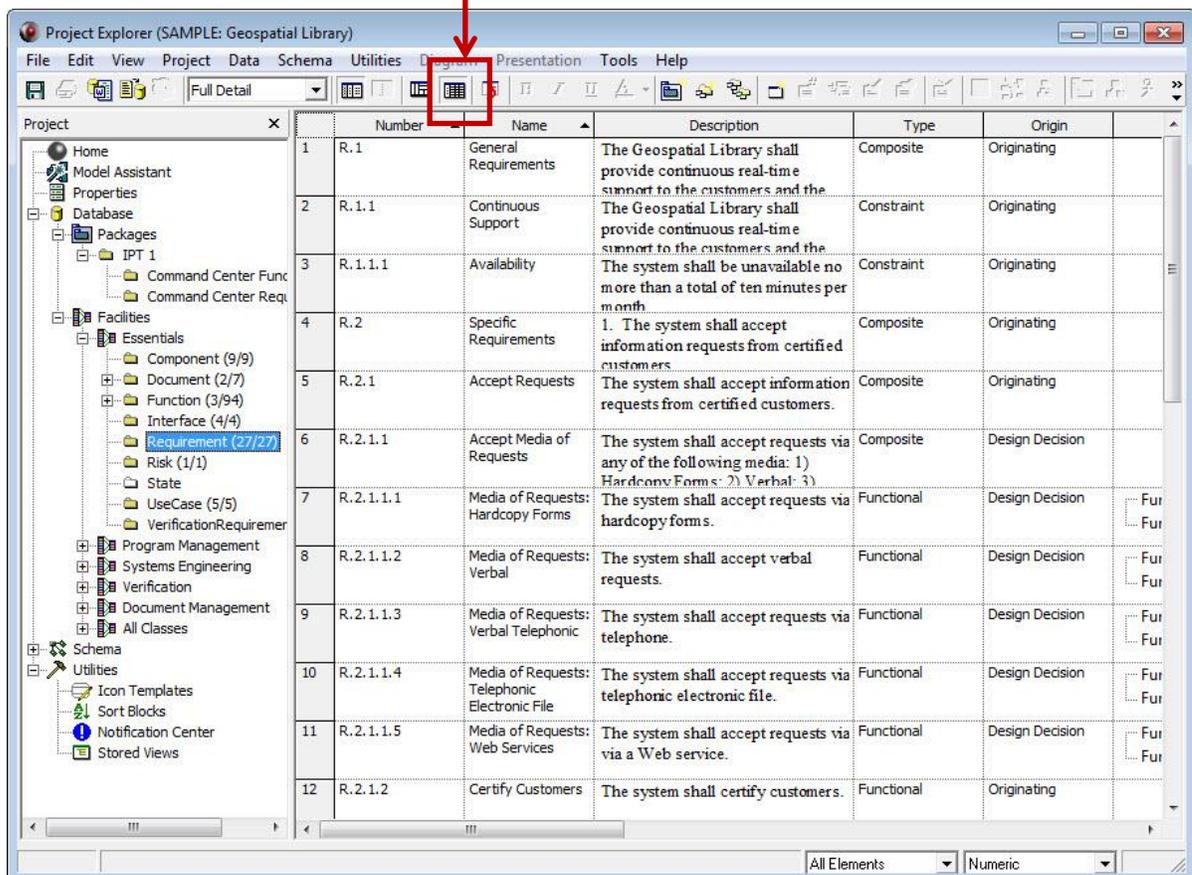
In addition to the Element Browser, CORE has an Element Table view that allows you to view, update, and add elements of a class in a spreadsheet-like presentation. The Element Table operates much the same way as Microsoft Excel – enter your information, reorder your columns, sort your data, resize as desired, etc. The Element Table can be viewed using the following steps:

- In the Project Explorer, select **Requirement** in the Project pane.
- On the toolbar, click the *Element Table*  icon. This will switch the Explorer display from the Element Browser to the Element Table view.
- Scroll through the rows and columns to become familiar with the layout of the Element Table window.

The attributes and relationships displayed for each class have been predefined. However, you can adjust this using the *View > Table > Edit Columns* command. Any changes will also apply to the Diagram Elements pane of the diagram views.

When using the Table view to work with elements, click in any cell in the element row. This will activate the view icons for that element. The Table View can also be used in conjunction with a diagram depending on the user's preference.

### Element Table



Number	Name	Description	Type	Origin
1	R. 1	General Requirements	Composite	Originating
2	R. 1.1	Continuous Support	Constraint	Originating
3	R. 1.1.1	Availability	Constraint	Originating
4	R. 2	Specific Requirements	Composite	Originating
5	R. 2.1	Accept Requests	Composite	Originating
6	R. 2.1.1	Accept Media of Requests	Composite	Design Decision
7	R. 2.1.1.1	Media of Requests: Hardcopy Forms	Functional	Design Decision
8	R. 2.1.1.2	Media of Requests: Verbal	Functional	Design Decision
9	R. 2.1.1.3	Media of Requests: Verbal Telephonic	Functional	Design Decision
10	R. 2.1.1.4	Media of Requests: Telephonic Electronic File	Functional	Design Decision
11	R. 2.1.1.5	Media of Requests: Web Services	Functional	Design Decision
12	R. 2.1.2	Certify Customers	Functional	Originating

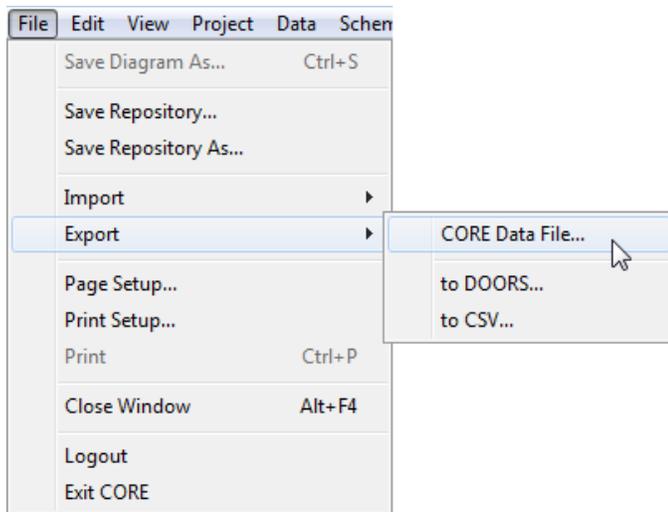
## Saving CORE Data

CORE uses XML as the primary format for all project specific data files. This provides “single file” storage for all project data: multiple projects, schema, stored views, sort, hierarchy, filters, etc. CORE project data imported and exported using XML will have an .xml file extension.

CORE University Edition users should go to the Appendix on page 64 to learn how to export/save CORE data.

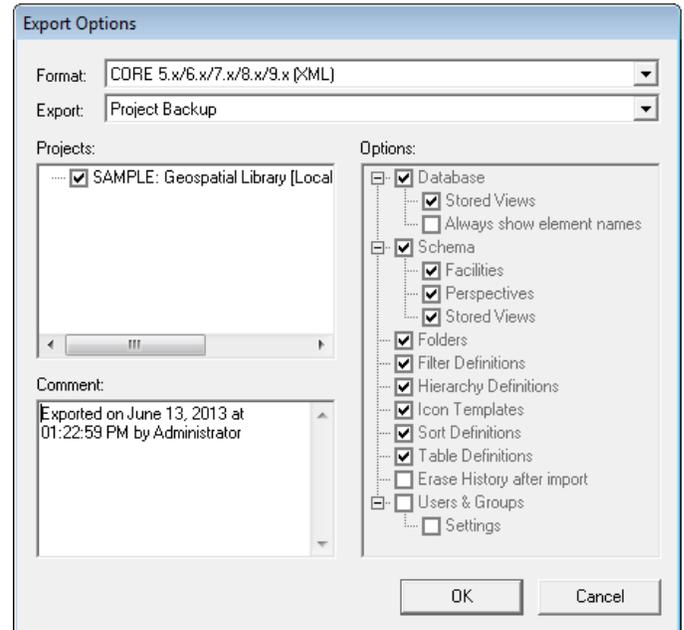
To save your CORE data to an XML file:

- Select **File > Export > CORE Data File** from the Project Explorer drop-down menus.

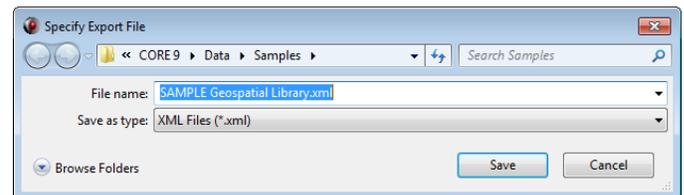


An Export Options dialog will open.

- Click **OK**, since we will use the default settings to export the entire project.

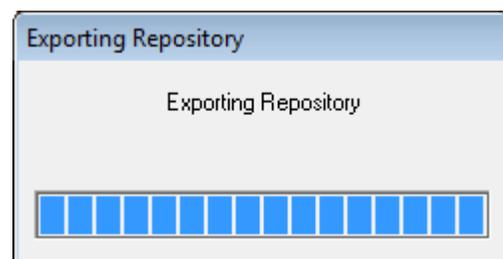


- When the Choose Export File dialog appears, name the export file **Geospatial Library.xml**.



- Click **Save**.

The Export Repository dialog indicates the progress of the export.



## Saving a CORE Repository

Now that you have seen how to import a database file and view data, in the next section, we will see how to build a CORE model. Let's start with an empty project.

- From the Project Explorer menu, select **Project > Erase > Database**.
- Answer *Yes* to the warning message.

As you build this model, you will want to save your work as you progress. In addition to saving your CORE data as we just did, you can save your CORE repository file to disk. A CORE repository file contains a copy of the CORE tool with all the projects and their schemas and data. Saving a CORE repository is the fastest way to save your work, but creates the largest data file. Exporting your data is more appropriate if you are creating a backup or are exchanging data with another user.

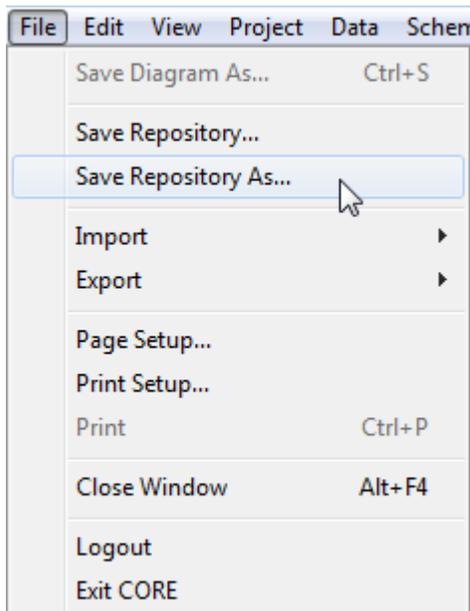
---

The ability to create a CORE repository file is not available in CORE University Edition. Users of this Edition are referred to the Exporting (Saving) a Data File section described in the Appendix on page 64 to save their work.

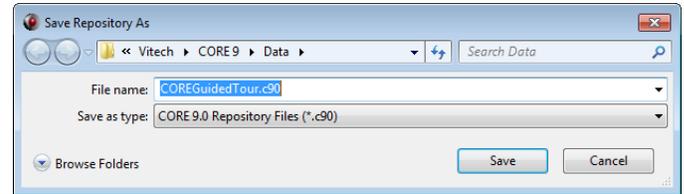
---

To save your CORE repository:

- Select **File > Save Repository As** from the drop-down menus.



- When the Save Repository As dialog appears, enter the file name **COREGuidedTour.c90** and click **Save**.



A Saving Repository dialog indicates activity while saving the repository.

From now on, as you incrementally create your model, you will be able to save your work by using the **File > Save Repository** command. This will overwrite the saved repository on disk with the current project information.

You have now become familiar with basic navigation, importing/exporting a project, and other commonly used features of CORE. It's time to create a sample problem.

### User Tip

It is recommended that you should save your data often. Saving the repository in CORE is a fast save (or exporting the database in the University Edition). Exporting the project in XML (not available in the University Edition) is the recommended format for backing up or exchanging your project information.

THIS PAGE INTENTIONALLY BLANK

# 3

## Starting a CORE Project

In this section, you'll begin with requirements

- Introduce the Guided Tour's User Exercise
- Extract Requirements into CORE
- Define Relationships
- View the Requirement Hierarchy

### The Sample Problem: Geospatial Library

In the rest of this guided tour, you will use CORE and several of its features by systematically building the sample design for the Geospatial Library. Please review the Geospatial Library context diagram and its description as found in the previous section (see page 5). You may find it helpful to refer back to this diagram as you build the system structure.

### Summary of the Typical Top-down Process

CORE supports top-down, bottom-up (reverse), and middle-out engineering processes/paradigms. For this guided tour, we are using a top-down approach assuming the Geospatial Library is an unprecedented system. We will start by extracting and analyzing the requirements from the source requirements documentation. Any system boundary and physical architecture constraints will be captured. Threads of requirements/functions are then developed to specify stimulus/response relationships required of the system to process each of the classes of system inputs. These threads are then combined into the integrated system logic, and the details of the functional requirement definitions are completed. These include all of the input/output (I/O) relationships, the processing steps, the I/O attributes, and the functional control logic as well as the functional requirement statements themselves.

The system definition language characterizes these requirements in the system design repository by capturing each requirement element's relationships and attributes. The developing system design model is concurrently analyzed by the design team for "fitness" as a solution and any inherent design impediments, while checking consistency and completeness. While the functional behavior definition is proceeding, the physical decomposition of the system is specified so that the behavior can be allocated to the physical system components. One result of this allocation is the definition

of all interfaces between the physical elements of the system, including hardware, software, and people. We will complete the following steps in this guided tour:

1. Capture the source document, often a starting point for top-down engineering.
2. Capture the requirements from the source documentation.
3. Define the system and its boundary.
4. Derive the system behavior while extending the physical architecture and allocating all behavior to the physical architecture.

After completing this, we will have established traceability among the relevant design elements, identified and resolved critical concerns, and provided documentation.

### Capturing the Problem and Source Requirements

In Section 2 we learned how to import and view an established project file. Now we will see how to build a CORE model starting with an empty project. **Before proceeding with this section, be sure to start with an empty project (see page 15).**

We will start by capturing the relevant information from the source document in CORE. CORE's Document Parser is an easy way to transfer text from a document into the CORE database structure.

- In the Project Explorer, click the *Document Parser* icon  or select the *Tools > Document Parser* command.

#### Note

In addition to the Document Parser, CORE also has an Element Extractor which allows you to import information from a source one section at a time. This method removes a layer of automation so that the user must specify each attribute of every element pulled from the source document. If you prefer this method you can use the Element

Extractor by pressing the icon .

Once the Document Parser opens we first need to import our source document.

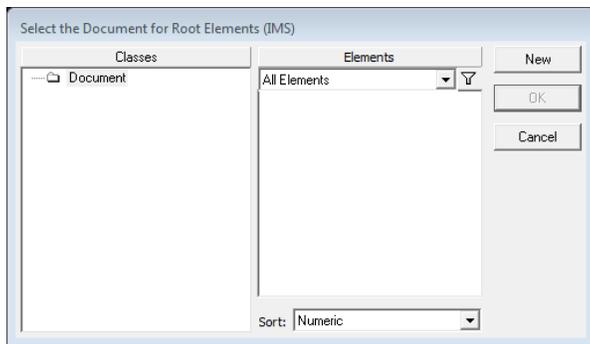
The source document must be either a DOC, RTF, HTML, or TXT file. To load your source document:

- Click on the **Load Document** button  or select **File > Load Document**.

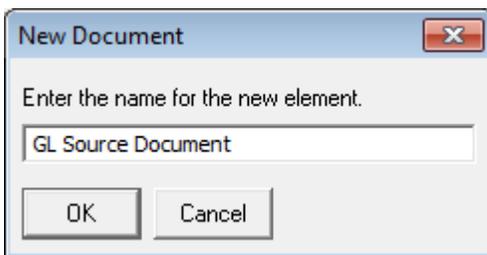
We will use the file GeospatialLibrarySourceDocument.doc, located in the Samples folder of the CORE directory.

- Navigate to the **CORE9\Data\Samples** subdirectory.
- Select **GeospatialLibrarySourceDocument.doc**.
- Click **Open**.

Next identify the source document that will be parsed by clicking **Select**.

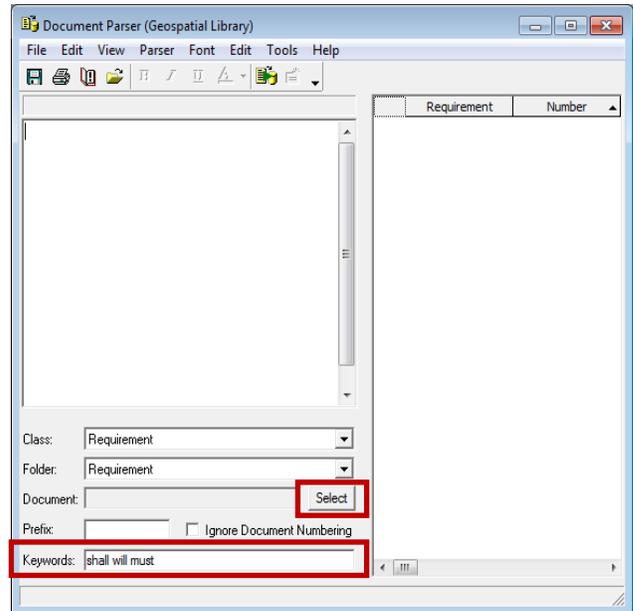


There are no documents currently in our model, so you need to create one. Select **New** to enter a new document. Enter **GL Source Document** then click **OK**.



Select the **GL Source Document** element then click **OK**.

Next you need to identify the Keywords (see figure below) that will be used in parsing the document. This field is auto-populated by default with the most common keywords that indicate a **Requirement**. Additional keywords may be added or deleted by the user.



**User Tip**

The CORE Icon Reference Guide is a two-page quick reference to the icons used in CORE. You can access it by selecting the **Help > Documentation** pull-down menu from the Project Explorer.

## Parsing the Document Element

The GeospatialLibrarySourceDocument.doc file displays in the left-hand pane of the Document Parser.

The Document Parser can extract text into any class of elements.

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

To parse the document

- Select the Document Parser icon  located in the Parser dialog box toolbar.

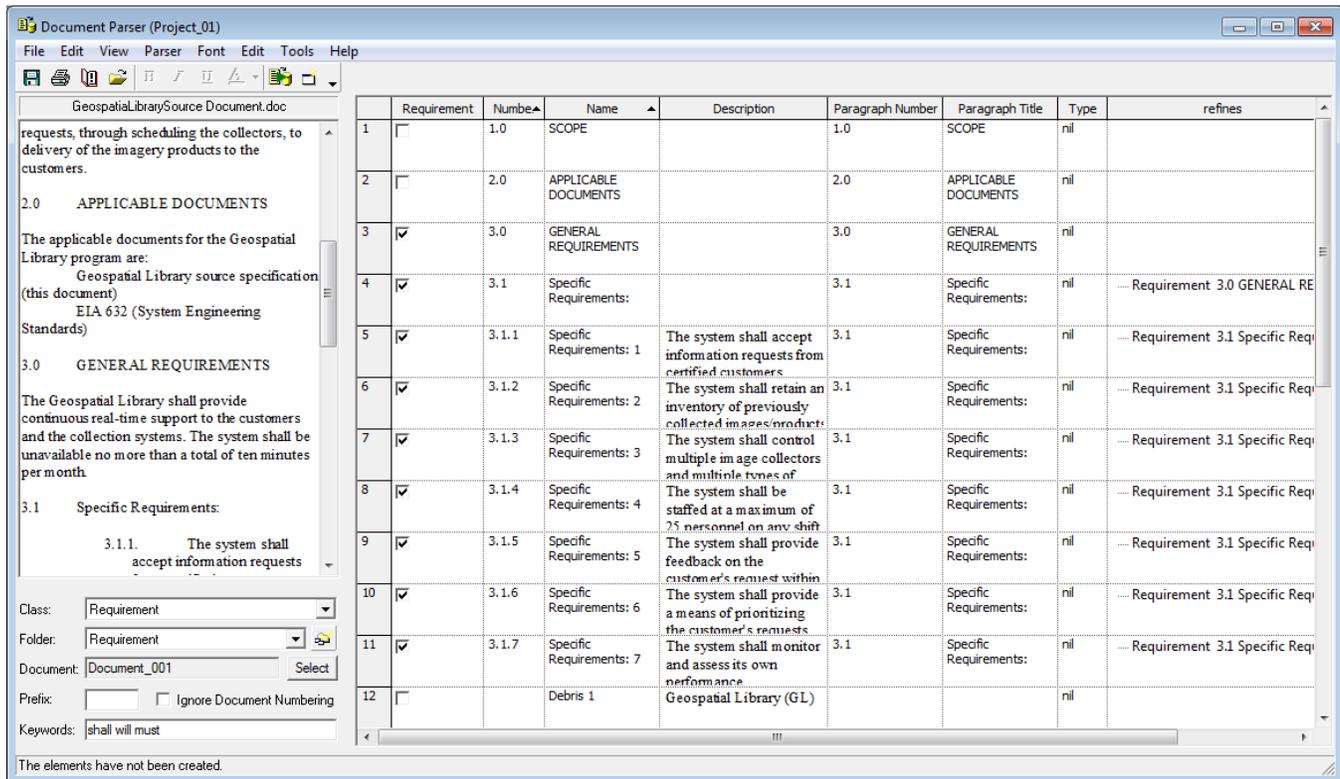
The Parser will then display the entire document's statements (or sentences) in the right-hand pane. Statements that contain one of the keywords are marked as **Requirements**. Other are loosely termed "debris" and may include boilerplate text or other valuable context. Where possible, the parser identifies attributes and the refines relationship for an element.

## Capturing the Elements into the Design Repository

Once all desired attributes are defined, we need to enter the **Document** element into the system design repository. If desired, you can edit the parsing results first in the table view of the Document Parser. In our case, we want to create all elements exactly as parsed.

- From the Document Extractor window, click the *Create Element* icon  on the toolbar to enter all of the requirements in the system design repository (or select *Extractor > Create Element*). The status line reflects when an element has been saved to the system design repository.

Close the Document Parser.



The screenshot shows the Document Parser interface with a table of requirements and a text pane on the left. The table has columns for Requirement, Number, Name, Description, Paragraph Number, Paragraph Title, Type, and refines. The text pane shows the document content with requirements highlighted.

Requirement	Number	Name	Description	Paragraph Number	Paragraph Title	Type	refines
<input type="checkbox"/>	1.0	SCOPE		1.0	SCOPE	nil	
<input type="checkbox"/>	2.0	APPLICABLE DOCUMENTS		2.0	APPLICABLE DOCUMENTS	nil	
<input checked="" type="checkbox"/>	3.0	GENERAL REQUIREMENTS		3.0	GENERAL REQUIREMENTS	nil	
<input checked="" type="checkbox"/>	3.1	Specific Requirements:		3.1	Specific Requirements:	nil	.... Requirement 3.0 GENERAL RE
<input checked="" type="checkbox"/>	3.1.1	Specific Requirements: 1	The system shall accept information requests from verified customers	3.1	Specific Requirements:	nil	.... Requirement 3.1 Specific Req
<input checked="" type="checkbox"/>	3.1.2	Specific Requirements: 2	The system shall retain an inventory of previously collected images/products	3.1	Specific Requirements:	nil	.... Requirement 3.1 Specific Req
<input checked="" type="checkbox"/>	3.1.3	Specific Requirements: 3	The system shall control multiple image collectors and multiple types of	3.1	Specific Requirements:	nil	.... Requirement 3.1 Specific Req
<input checked="" type="checkbox"/>	3.1.4	Specific Requirements: 4	The system shall be staffed at a maximum of 25 personnel on any shift	3.1	Specific Requirements:	nil	.... Requirement 3.1 Specific Req
<input checked="" type="checkbox"/>	3.1.5	Specific Requirements: 5	The system shall provide feedback on the customer's request within	3.1	Specific Requirements:	nil	.... Requirement 3.1 Specific Req
<input checked="" type="checkbox"/>	3.1.6	Specific Requirements: 6	The system shall provide a means of prioritizing the customer's requests	3.1	Specific Requirements:	nil	.... Requirement 3.1 Specific Req
<input checked="" type="checkbox"/>	3.1.7	Specific Requirements: 7	The system shall monitor and assess its own performance	3.1	Specific Requirements:	nil	.... Requirement 3.1 Specific Req
<input type="checkbox"/>		Debris 1	Geospatial Library (GL)			nil	

The elements have not been created.

## Identifying Other Elements

Our next step is to identify other useful statements that came in with the Requirements and transform them to the appropriate class.

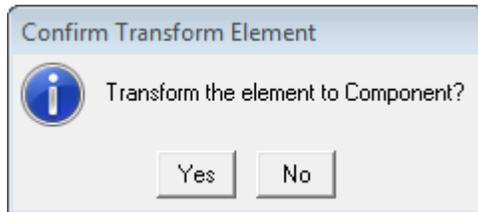
- In the Project Pane, select the **Requirement** class.

There are 24 elements listed that were created by the Document Parser.

Debris 0003 contains a description of the Geospatial Library and would therefore serve as a basic element for constructing a Component Element.

To transform an element from one class to another:

- First switch back to Element Browser view using the Element Browser icon. 
- Click and hold on **Requirement Debris 0003**.
- Drag **Debris 0003** over to the Project Pane and drop it on the **Component** class.
- The following dialog will appear. Select *Yes* to confirm.



- Click on the **Component** class in the Project Pane.
- Then select the **Debris 0003** element.

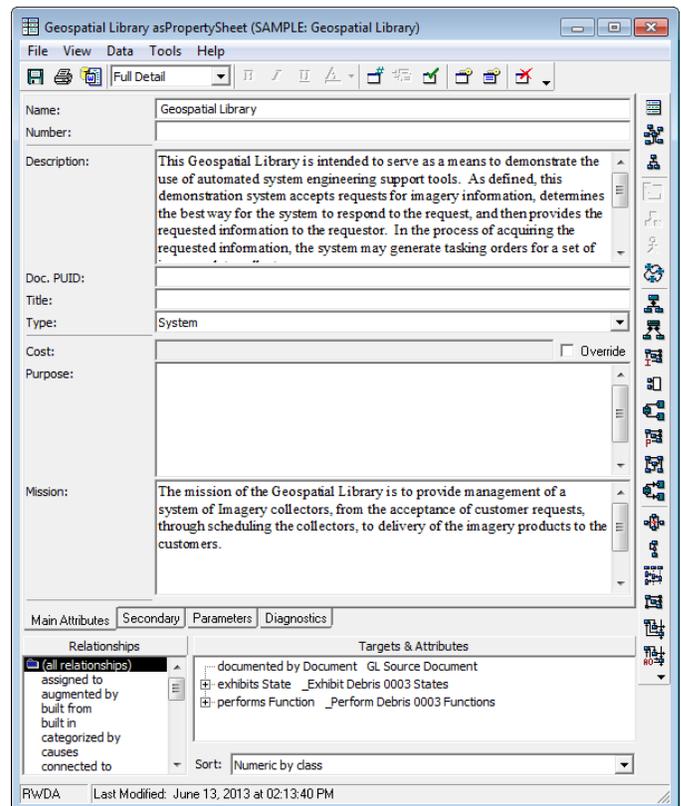
The Property Sheet of the Debris 3 element will be displayed. Let's complete the definition of this element.

- Click in the **Name** field and replace the current text with **Geospatial Library**.
- Set the **Type** attribute to **System** using the drop down selection list.

There is another “Debris” element in the **Requirement** class that will help here.

- Switch back to the **Requirement** class and select **Debris 0007**.
- Highlight the text in the **Description** field, right-click and select *Copy*.
- Switch back to the **Component** class and select the **Geospatial Library** element.
- Right-click in the **Mission** field and select *Paste*.

Compare your property sheet to the one shown below. Notice that there is still a relationship that needs to be established in order to link this element back to the source document.

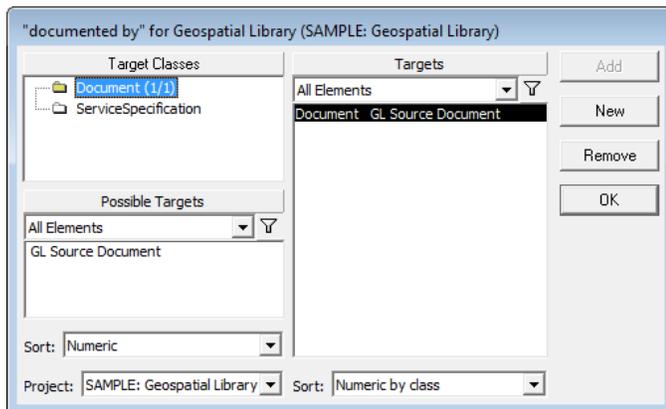


## Defining a Relationship

For traceability, we want to establish relationships between various elements. The Document Parser automatically established refines relationships between the **requirements** and documented by relationships between the top level **requirements** and the GL Source Document element.

The following step will establish a documented by relationship between the **Component** Geospatial Library and the GL Source Document element.

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



The Edit Targets dialog is used to add one or more targets to the selected relationship (or to remove targets from the selected relationship).

- Select the **GL Source Document** element in the Possible Targets pane.
- Click *Add*, then *OK*.

The remaining **requirements** should be reviewed to see what other useful information has been captured. Debris 0002 contains a **description** statement for the GL Source Document.

- Navigate to the **Debris 0002 Requirement**.
- Copy the **description**.
- Switch to the **Document** class and select the **GL Source Document**.
- Paste the text into the **Description** field.

Debris 0010 can serve as a reference document.

- Switch back to the **Requirement** class and select the **Debris 0010** element.
- Drag the **Debris 0010** element onto the **Document** class folder.
- Click *Yes* to transform the element.
- Switch to the **Document** class and select the **Debris 0010** element.
- Rename the element **EIA 632**.
- Establish a referenced by relationship with the **GL Source Document** element.

The Document Parser created a numbering scheme from what it found in our Source Document. Although it named our Specific Requirements elements, we need more detail for these names to be meaningful when viewed in a list.

Let's first switch to Element Table View to view the Requirements class in a table. Table View is a very efficient way to accomplish the bulk editing we are about to do.

- Click the *Table View* icon  in the Toolbar, then select the **Requirements** class.
- Take a moment to examine the **description** attributes of the elements with "Specific Requirements" in their names. It's easy to see what their names should be.
- Click in the **name** field of element number 3.1.1.
- Replace the current text with **Accept Requests**.
- Repeat this process to rename all of the Specific Requirements as outlined in the table on the next page.

	Number ▲	Name ▲
1	3.1	Specific Requirements:
2	3.1.1	Accept Requests
3	3.1.2	Retain Inventory and Provide Products
4	3.1.3	Control Multiple Collectors and Collector Types
5	3.1.4	Maximum Staff
6	3.1.5	Provide Feedback
7	3.1.6	Prioritize Requests
8	3.1.7	Monitor and Assess Performance

Notice the two General Requirements at the bottom of your Requirements list. These need to be renamed as well as numbered.

- Rename General Requirements 1.1 to **Continuous Support** and set the **number** to **3.0.1**.
- Rename General Requirements 1.2 to **Availability** and set the **number** to **3.0.2**.

This concludes the review of the elements created by the Document Parser. We can now delete the unused elements using one of the following methods:

- First, switch back to Element List view using the Element List icon  in the Toolbar.
- Using the **Ctrl + Click** method, multi-select all of the remaining “Debris” **Requirements, General Requirements** (from the bottom of the list), **Scope**, and **Applicable Documents**.
- Click the *Delete Element* icon  or click **Data > Delete > Elements** from the menu.

The following shows the resulting list of requirements listed in the elements pane.

```

3.0 GENERAL REQUIREMENTS 1
3.0.1 Continuous Support
3.0.2 Availability
3.1 Specific Requirements:
3.1.1 Accept Requests
3.1.2 Retain Inventory and Provide Products
3.1.3 Control Multiple Collectors and Collector Types
3.1.4 Maximum Staff
3.1.5 Provide Feedback
3.1.6 Prioritize Requests
3.1.7 Monitor and Assess Performance

```

## Establishing Traceability to the Source Document

Now that we have established our requirements we need to establish traceability to our GL Source Document.

The Document Parser took care of the refines relationships that link second-level requirements to their parent requirement, but because of the structure of the document, it did not connect the two primary parent requirement elements to the **GL Source Document**.

We have two primary parent **requirement** elements: **General Requirements** and **Specific Requirements**. General Requirements already has a documented by relationship that traces to the source documents. We need to establish the same relationship for **3.1 Specific Requirements**.

- Select **General Requirements**.
- Establish a documented by relationship with **GL Source Document** as the target.
- Repeat for 3.1 Specific Requirements.

Finally, we need to set the Type attribute for all of our **Requirements**. This will prepare us for the next step in our process where we will extract child-level **requirements** by identifying the composite **requirements**.

- Edit the **Type** attribute for all **Requirements** as defined in the table below.

	Number ▲	Name ▲	Description	Type
1	3.0	GENERAL REQUIREMENTS 1	The Geospatial Library shall provide continuous real-time support to the customers and the collection systems. The	Composite
2	3.0.1	Continuous Support	The Geospatial Library shall provide continuous real-time support to the customers and the collection systems.	Constraint
3	3.0.2	Availability	The system shall be unavailable no more than a total of ten minutes per month.	Constraint
4	3.1	Specific Requirements:		Composite
5	3.1.1	Accept Requests	The system shall accept information requests from certified customers.	Functional
6	3.1.2	Retain Inventory and Provide Products	The system shall retain an inventory of previously collected images/products and provide them to users, if appropriate.	Composite
7	3.1.3	Control Multiple Collectors and Collector Types	The system shall control multiple image collectors and multiple types of image collectors.	Composite
8	3.1.4	Maximum Staff	The system shall be staffed at a maximum of 25 personnel on any shift.	Constraint
9	3.1.5	Provide Feedback	The system shall provide feedback on the customer's request within twenty four hours.	Performance
10	3.1.6	Prioritize Requests	The system shall provide a means of prioritizing the customer's requests.	Functional
11	3.1.7	Monitor and Assess Performance	The system shall monitor and assess its own performance.	Composite ▼

Take a moment to review your elements before moving on.

This is also a good time to save your work!

## Extracting the Child-Level Originating Requirements

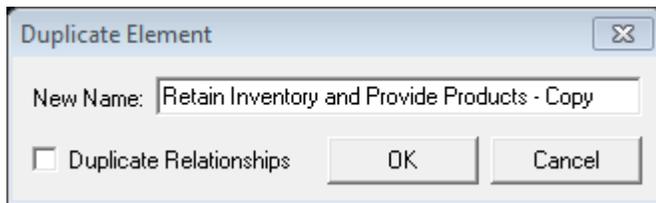
Many of the captured **requirements** are compound statements. We will break out the individual statements into child-level **requirements**. Each child **requirement** refines its parent. Traceability back to the source document, **GL Source Document**, is achieved through the parent (as defined with the documented by relationship).

Let's switch back to List View for this work.

- Click the Element List icon  in the Toolbar.

The *Data > Duplicate Element* command will allow you to create a copy of an element which is particularly useful when creating child level elements.

- Select **3.1.2 Retain Inventory and Provide Products**.
- Click *Data > Duplicate Element*.



The Duplicate Relationships option on this dialog will allow you to copy the element with or without relationships.

- Click in the box next to Duplicate Relationships to remove the check mark (this will duplicate the element without the relationships)
- Edit the New Name field to read **Retain Inventory**.
- Click *OK*.

The Element Browser will now display the Property Sheet of the new element

- Set the **Number** attribute to **3.1.2.1**.
- Set the **Type** attribute to **Functional**.
- Create a refines relationship with **3.1.2 Retain Inventory and Provide Products** as the target.

Now we'll repeat this process to create the remaining child-level **requirements** using the list below.

- Duplicate element 3.1.2
  - **Name:** Provide Products
  - **Number:** 3.1.2.2
  - **Type:** Functional
  - Refines: 3.1.2
- Duplicate element 3.1.3
  - **Name:** Control Multiple Collectors
  - **Number:** 3.1.3.1
  - **Type:** Functional
  - Refines: 3.1.3
- Duplicate element 3.1.3
  - **Name:** Control Multiple Collector Types
  - **Number:** 3.1.3.2
  - **Type:** Functional
  - Refines: 3.1.3
- Duplicate element 3.1.7
  - **Name:** Monitor Self Performance
  - **Number:** 3.1.7.1
  - **Type:** Functional
  - Refines: 3.1.7
- Duplicate element 3.1.7
  - **Name:** Assess Self Performance
  - **Number:** 3.1.7.2
  - **Type:** Functional
  - Refines: 3.1.7

## Viewing a Requirements Diagram

Now that the child-level **Requirement** elements have been created, let's view a Requirements Diagram from our top-level **requirement**.

- In the Elements Pane, select **3.1 Specific Requirements** in the Project pane.
- Click the *Requirements Diagram* icon  from the toolbar to open a Requirements Diagram following the SysML diagram convention.

Your diagram should look similar to the one shown below.

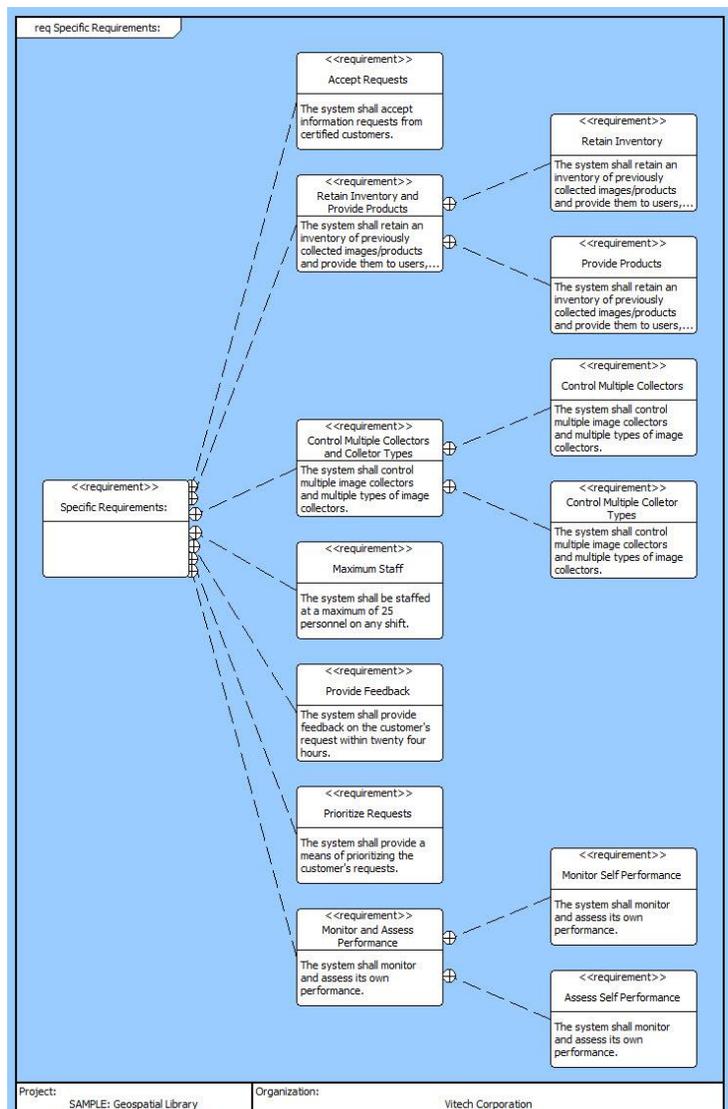
In a few short steps we have loaded the project source material. We have done this with a human in the loop to ensure both a first-level understanding and that we didn't lose valuable context in the translation from a document to a requirements model. Real-world projects will take longer than our sample, however the rapid time-to-first-value will help your project team get started quickly.

**NOTE:**  
CORE Essentials Users can perform the steps on this page in the Hierarchy Diagram.

### Note

In order to better fit the page, we took advantage of options to adjust the layout of the diagram. To change the layout, in the toolbar, select the Set

Layout button . Then select the desired layout. We used the horizontal layout in the image below.



## Viewing a Traceability Spider or Hierarchy Diagram

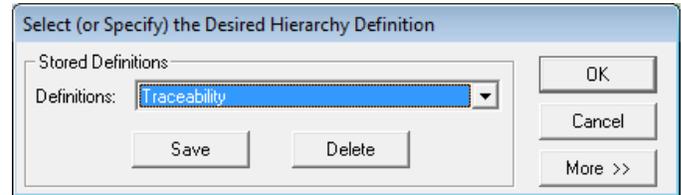
In addition to the SysML Requirements Diagram, let's look at a Tracability view in either a Spider Diagram 

or the Hierarchy Diagram  from our source document. If we select the GL Source document, we can show traceability from our source document down through the **requirements**. As our model progresses, the diagram can be expanded to show traceability through the entire design.

Recall that our **Requirements** are documented by our source Document named GL Source Document. This means that our **Document documents** our Originating Requirement.

- In the Project Explorer, select **Document** in the Project pane.
- Select the GL Source Document element show the property sheet and to activate the view icon tabs at the bottom of the property sheet.
- Click on either the *Spider Diagram* icon  or the *Hierarchy Diagram* icon  from the toolbar to /open a Hierarchy Definition Dialog.

The Hierarchy Definition Dialog provides selections for building various hierarchy diagrams. You can click the drop-down arrow to see the default set of hierarchy diagrams available. We will use **Traceability** which was selected by default.

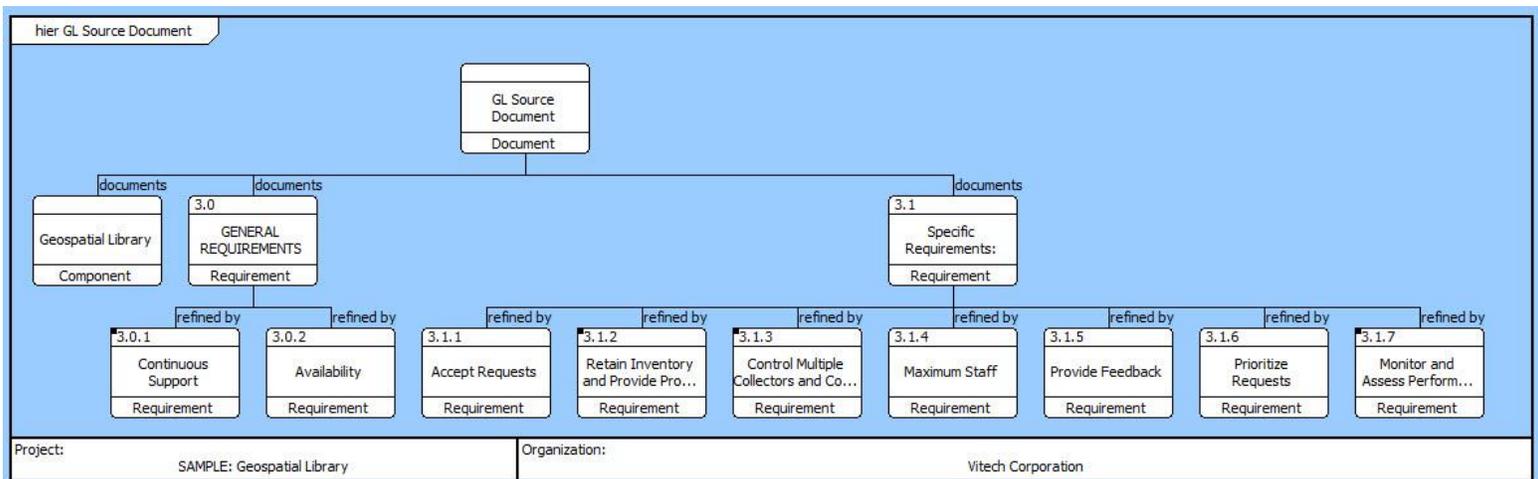


- Click **OK**.

Since the GL Source Document was the source from which we extracted the specific **requirements**, we have traceability from the source document through the second-tier (child-level) **requirements**.

Use the Layout icon  to change the layout of the diagram.

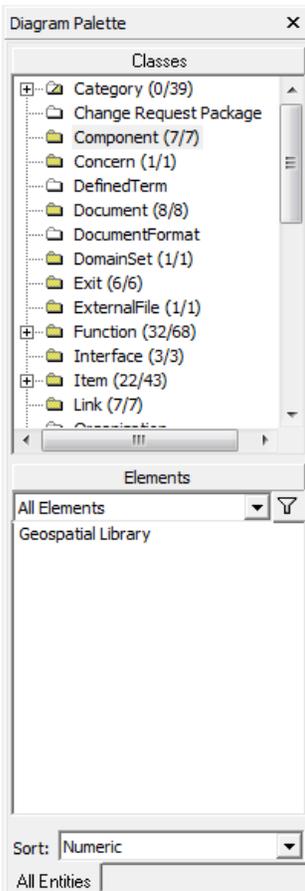
Below is a Hierarchy Diagram using the Vertical layout.



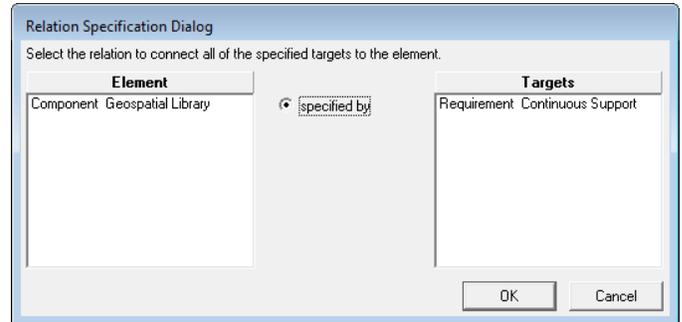
## Adding Elements in a Traceability Spider Diagram

CORE allows us to create elements and relationships directly in a diagram. Using the Spider Diagram, we will create a specifies relationship for **Continuous Support**.

- If you previously opened a Hierarchy Diagram, close that window, then click on the Spider Diagram icon selecting the Traceability view. Then set the layout to vertical.
- From the Spider Diagram's Diagram Palette select the All Entities tab, then click on the **Component** class, then the element **Geospatial Library**.



- Drag **Geospatial Library** on top of **Continuous Support**, dropping it there to open the Relation Specification Dialog.

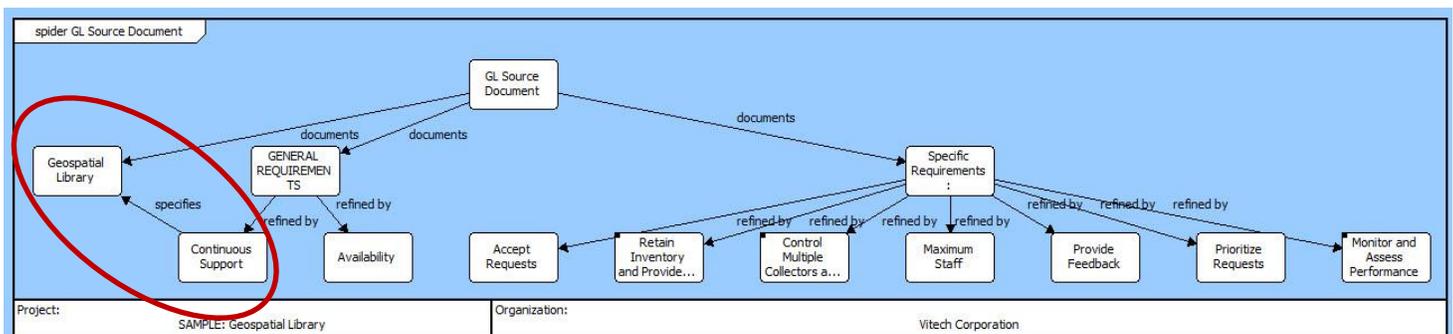


- Select **OK**.

Notice in the diagram at the bottom of the page that new relationships have been added to the appropriate elements.

### NOTE

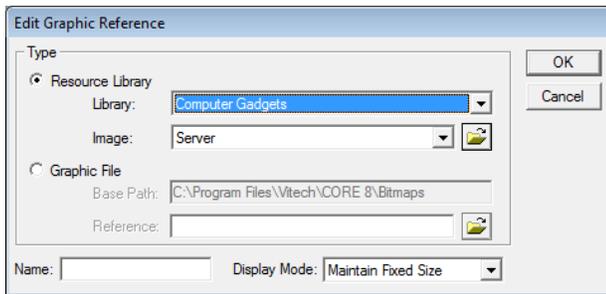
A single icon with a black dot in the upper-left corner can be expanded individually. Select the icon, and then right-click to view a menu. Select **Presentation** from the menu, then select **Expand Nodes** command and set the number of levels you which to expand below the highlighted icon, the default is one. An icon can also be expanded/collapsed by using the plus/minus icons on the diagram toolbar or holding down the **Ctrl** key and double-clicking the icon to expand/contract one level at a time.



## Replacing Icons with Graphics

CORE 9 has a library of graphics that can be used to enhance the visualization of your diagrams. Let's set the **Geospatial Library** element to display using one of these images in place of the traditional icon.

- Select the Geospatial Library icon in the drawing.
- Click on the Set Image icon  in the diagram Toolbar.

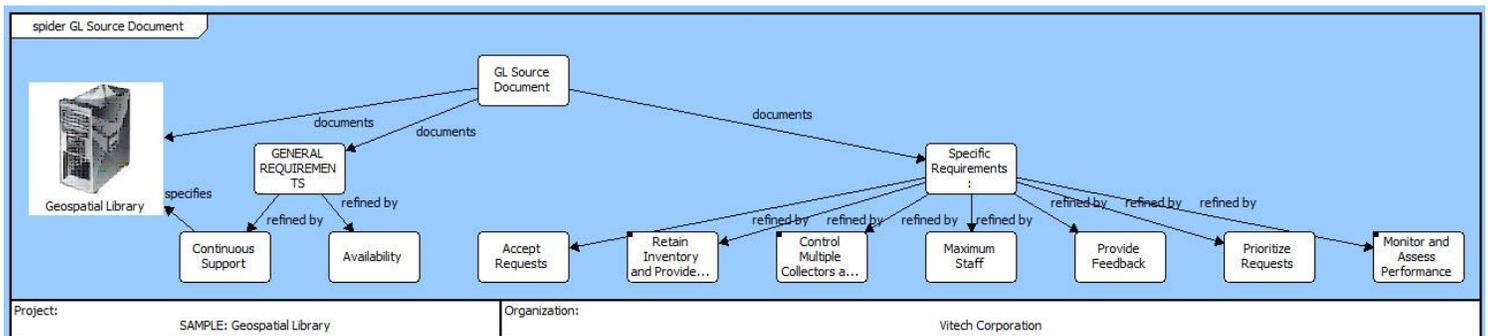


Note the two primary options in the Edit Graphics Reference box. Resource Library will allow you to select from the library that installed with CORE 9. The Graphic File option allows you to select from other images on your computer.

- Select **Resource Library**.
- Set the Library option to **Computer Gadgets**.
- Select **Server** in the Image field, then click **OK**.
- Finally, resize the image display area by clicking on the image and dragging the handle to the desired size.

Your diagram should look similar to the one below.

- Close the Spider Diagram.



## Capturing Requirements Concerns

Now that we have extracted the source **requirements**, requirement analysis begins. As a systems engineer, you want to identify problems encountered during systems engineering such as poorly stated or conflicting requirements. In CORE, these problems can be captured as **Concerns**. A **Concern** is an element in the design repository that identifies a problem (as well as its resolution). The primary application for the **Concern** class is documenting problems with **requirements** but may be used in conjunction with any class/element in the repository.

In our set of source **requirements**, we can see that our system must accept requests from the customers. However, the format of the requests is not identified.

Let's capture that **Concern**.

Take a look at the Project Pane. Did you notice that **Concern** is not listed in our list of classes?

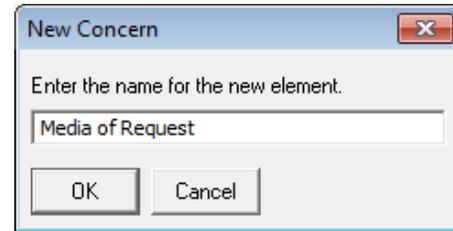
CORE has five defined Facilities listed in the Project Pane. A Facility is a group of classes chosen for an area of interest. By default, CORE opens to the Essentials facility when the project is first opened. This contains the foundational root classes for systems engineering and doesn't include the **Concern** class.

Let's switch to the Systems Engineering facility whose classes have been chosen because their elements are associated specifically with system design and specification.

- Click the *minus* next to Essentials to collapse that facility.
- Click the *plus* next to Systems Engineering to display that facility.

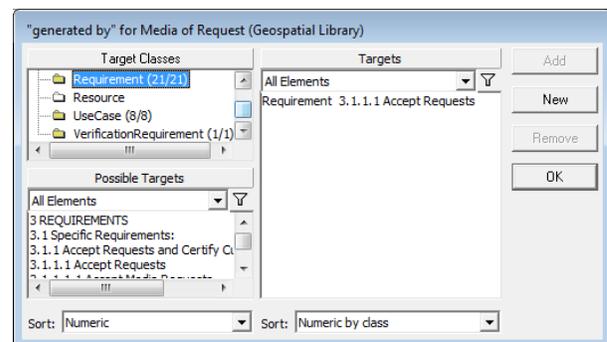
Now, let's get back to capturing our **Concern**.

- Select the **Concern** class.
- Set the Project Explorer to the Element List view by either clicking on the *Element Browser* icon  or by selecting *View > Element Browser*.
- In the Project pane, double-click **Concern** to create a **Concern** element.
- Name the element **Media of Request** and press *OK* to close the dialog.



We link a **Concern** to the element that generated the problem via the generated by relationship.

- Double-click generated by in the Relationships pane. Add the **Requirement 3.1.1 Accept Requests** to the Target list.
- Click *OK*.

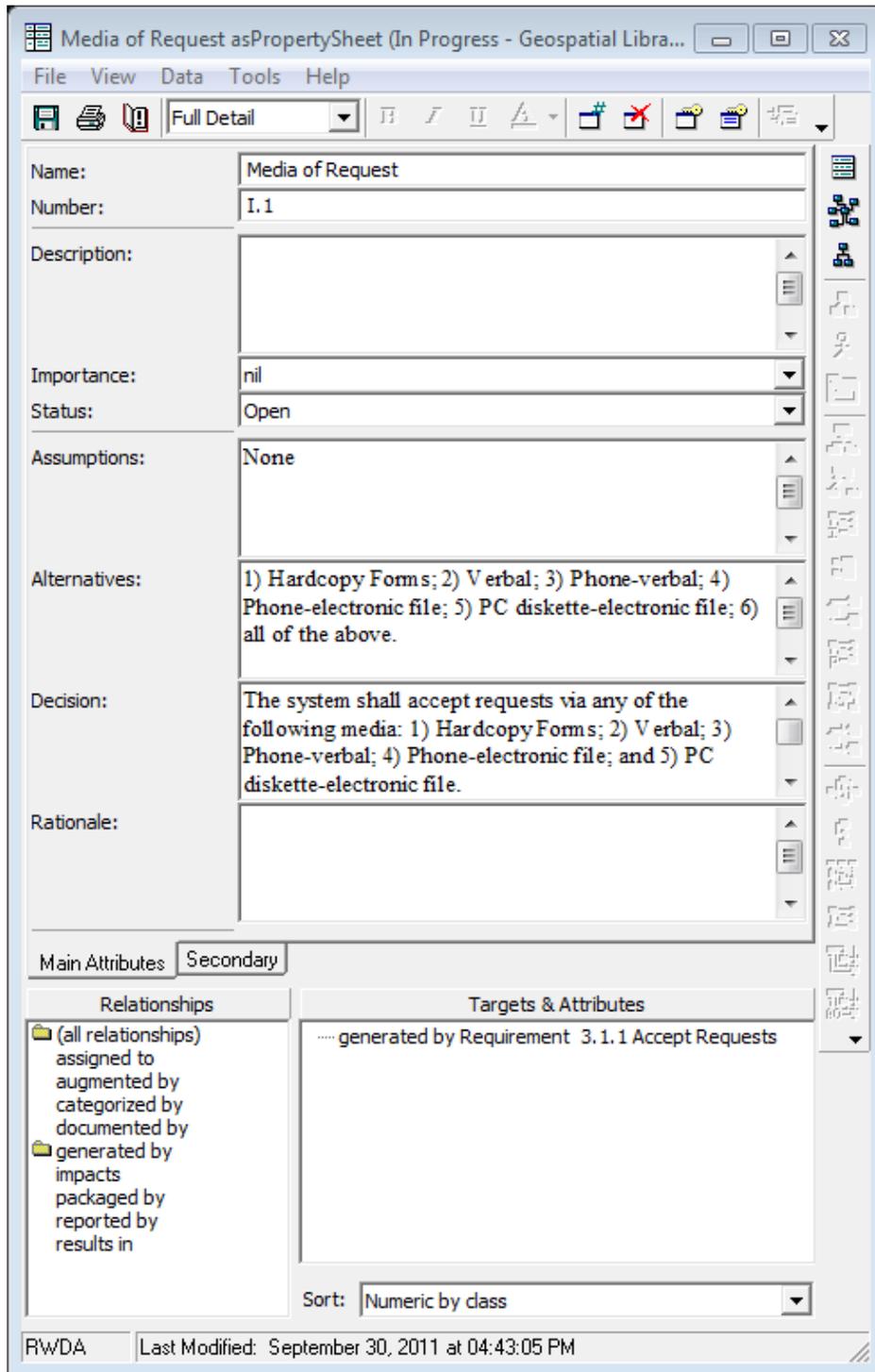


CORE allows you to capture both the concern and the decision that resolves the concern. You can document your **decision**, your **alternatives**, and your **rationale**. In this way, CORE serves as a repository for the project design history—capturing “the why” of systems engineering decisions.

- We will complete the Media of Request element to capture the analysis of the problem. Type text in the *Description* and *Alternatives* fields. You can use the text as shown on the next page, or develop your own.

### User Tip

Now is a good point to save your work by selecting *File > Save Repository* from the Project Explorer toolbar. In the future, don't forget to save your work periodically or whenever you need to stop and come back to the Guided Tour at a later time.



Media of Request asPropertySheet (In Progress - Geospatial Libra...)

File View Data Tools Help

Full Detail

Name: Media of Request

Number: I. 1

Description:

Importance: nil

Status: Open

Assumptions: None

Alternatives: 1) Hardcopy Forms; 2) Verbal; 3) Phone-verbal; 4) Phone-electronic file; 5) PC diskette-electronic file; 6) all of the above.

Decision: The system shall accept requests via any of the following media: 1) Hardcopy Forms; 2) Verbal; 3) Phone-verbal; 4) Phone-electronic file; and 5) PC diskette-electronic file.

Rationale:

Main Attributes Secondary

Relationships

- (all relationships)
- assigned to
- augmented by
- categorized by
- documented by
- generated by
- impacts
- packaged by
- reported by
- results in

Targets & Attributes

..... generated by Requirement 3.1.1 Accept Requests

Sort: Numeric by class

RWDA Last Modified: September 30, 2011 at 04:43:05 PM



# 4

## Defining the System and its Boundary

In this section, you define your system

- Create the System
- Establish the System Boundary
- View the Physical Hierarchy
- Define Top-Level Use Case

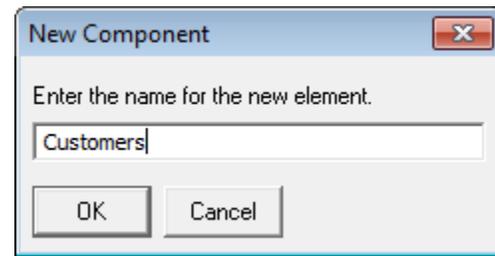
### Defining the System Environment

In order to define the system and its 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 what is outside of our system.

A **Component** element of **Type System** is used to identify the system and capture the system-level mission. A **Component** is an abstract term that represents the hardware, software, people, or grouping thereof that performs a specific function or functions.

In order to define the system's physical environment we will create the **Physical Context Component** element.

- From the Project Explorer, select the **Component** class and create a new element **Physical Context**.
- Click the *Structure BDD* tab .
- From the Diagram Palette, click the Key Entities tab then drag the **component Geospatial Library** on top of **Physical Context** and create the built in relationship.
- Click on the Constructs tab in the Diagram Palette.
- Hold down CTRL while dragging the New Node construct and dropping it onto the Physical Context element. (Hold CTRL until after you have dropped the construct.)
- Name the new element named **Customers**.



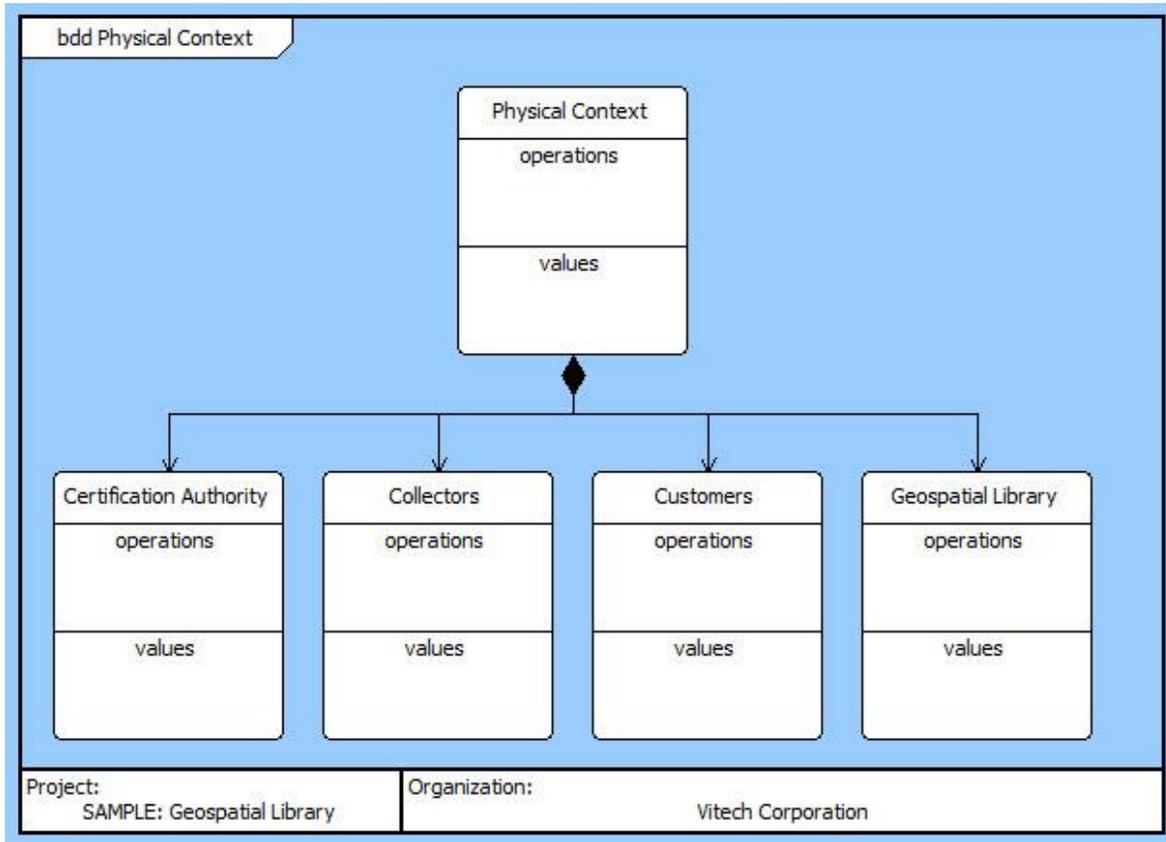
- This process creates the element and establishes the built in relationship.
- Repeat adding two more **components Collectors**, and **Certification Authority**, establishing built in relationships to the Physical Context element.
- Compare your diagram to the one shown on the next page.

In the spirit of building a well defined model, let's now complete the definitions of the elements we just created. Element Table View is a great tool for this type of bulk editing.

- Click the Table View icon .
- Using the table below, complete the definitions of the elements in Table View.
- When you are done, switch back to Element List view.

#### NOTE

CORE Essentials users can complete the steps on this page in the Physical Block Diagram. Drop the New Node icon on the diagram background instead of on an element.



	Number▲	Name ▲	Description	Type
1	C	Physical Context	The Context is the supersystem that shows the physical hierarchy relationship between the Geospatial Library and the external systems: Customers and Collectors.	Context
2	C.1	Customers	The external system, customers, prepare requests for imagery products.	Human
3	C.2	Collectors	The external system, collectors, is responsible for acquiring data to support the information that is requested but does not reside in the inventory. There are multiple collectors, of different types and capabilities.	External System
4	C.3	Certification Authority	Provides the capability to verify a user's certification to use the Geospatial Library.	External System
5	SYS.1	Geospatial Library	This Geospatial Library is intended to serve as a means to demonstrate the use of automated system engineering support tools. As defined, this demonstration system accepts requests for imagery information, determines the best way for the system to respond to the request, and then provides the requested information to the requestor. In the process of acquiring the requested information, the system may generate tasking orders for a set of imagery data collectors.	System



## Identifying System Level Interactions

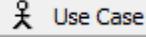
### NOTE

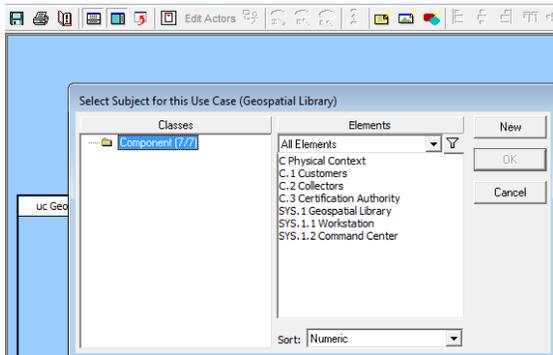
CORE Essentials should skip this section. There is no equivalent diagram and this capability is unique to CORE Spectrum.

Next, we begin the behavior analysis of our system by creating a Use Case Diagram. Here we will show the view. This Use Case will bridge the operational concepts, system boundary and requirements with the system behavior that brings the system to life.

Our source document identified how the Geospatial Library interacts with the Customers and Collectors. We will use this guidance to define our use case.

We first need to establish our parent Use Case element which will serve as a container for the use cases we want to address.

- In the Project Explorer create the **UseCase Operate Geospatial Library** and open a Use Case Diagram, using the *Use Case* tab .
- Start by identifying the subject for the Use Case by selecting the *Subject* icon  and the **component Geospatial Library**.

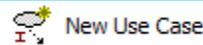


This creates a describes/decribed by relationship.

An analysis of our **requirements** will reveal two primary Use Cases that our system must address: delivering images and assessing performance.

Let's place these in our container.

- Using the Diagram Palette on the right side of the screen, drag a *New Use Case* construct



onto the diagram.

- Repeat this to add a second Use Case.

We now have two generic elements that must be defined.

- Right click on **UseCase\_001** and select *Rename Element*.
- Name the element **Deliver Images**.
- Repeat with **UseCase\_002**, renaming it **Assess Performance**.

Now let's consider what actors will be involved. Actors include both human and non-human elements. Because our systems purpose is to deliver images, Customers are a vital actor.

Actors on Use Cases come from the **Component** class and the elements that we've already defined.

- Select the **Deliver Image** element.
- Click the *Edit Actors* icon .
- Add **Customers**.

### Note

This element was inserted on our diagram as a stick figure. When we previously created this **component** we set the **Type** attribute to Human. That attribute selection will result in a stick figure being represented here.

We also know from our source document that the Geospatial Library will be the source of the images so we can add this element as the actor that delivers images.

- From the Diagram Palette, drag the *Actors* icon  and drop it on the **Deliver Images** element.
- In the "involves" relationship dialog, select **Geospatial Library**, click *Add*, then *OK*.

Take a look at the **Assess Performance** Use Case. This back office task doesn't involve the Customer, but certainly involves the Geospatial Library. Let's add that interaction.

- Hold down the **Ctrl** key, then drag the **Geospatial Library** actor and drop it on the **Assess Performance** Use Case.

- The Relation Specification Dialog is auto-populated with relationships that are valid between these two elements. Select participates in and click **OK**.

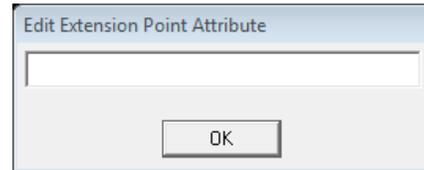
Now that we've defined our Use Case at a very high level, it's easy to see variant paths of delivering images to the customer. It's possible that the system may deliver an image that it already has, and it's also possible that the customer may have requested an image not in the system's image library and one will need to be requested from the Collectors.

Both of these situations are extensions of the Deliver Images Use Case.

- Select the **Deliver Images** Use Case.
- Click on the **Extend** icon .
- Click **New**, then name your new element **Retrieve Existing Product**.
- Click **New** again, and name the element **Collect and Deliver New Product**.
- Click **OK** to close the dialog.

Finally, we need to add a few notations to clarify the Use Case. Specifically, we need to note whether an image is delivered through the Retrieve Existing Product element, or through the Collect and Deliver New Product element.

- Select **Retrieve Existing Product**.
- Click the **Specify Extension Point** icon .

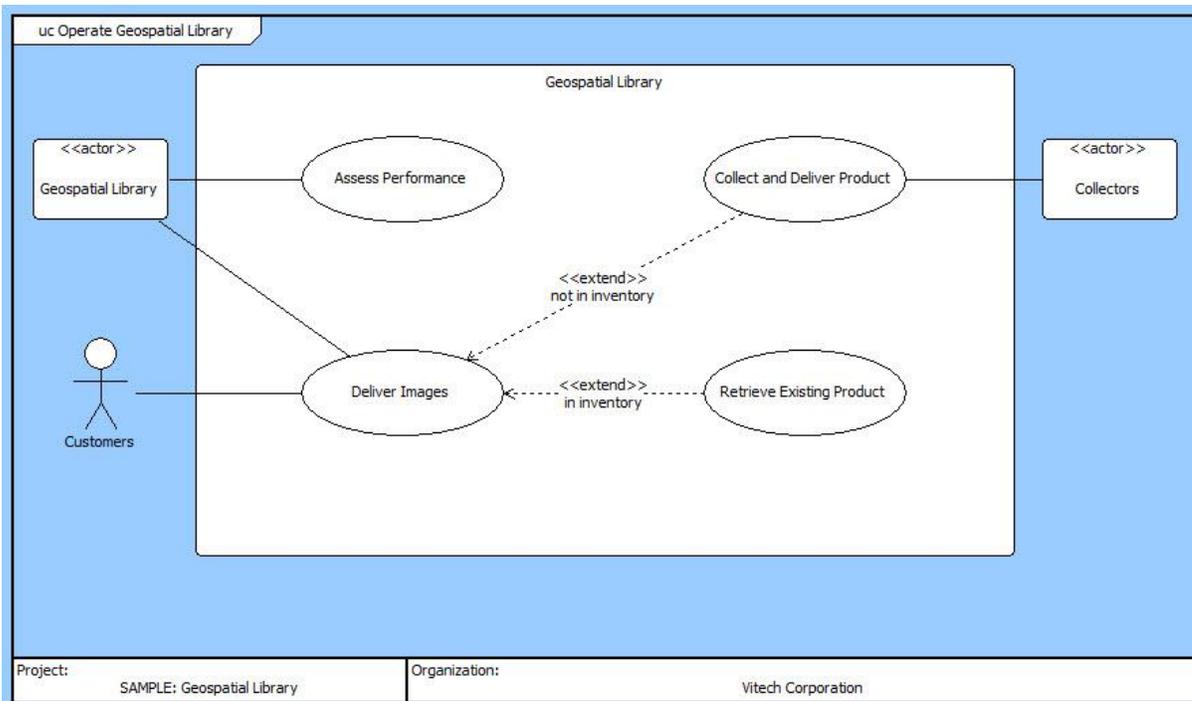


- In the Edit Extension Point Attribute dialog, enter **in inventory**.
- Repeat this process for the **Collect and Deliver New Product** element, defining the attribute as **not in inventory**.

The Geopspatial Library is a repository of images and delivers the images it stores. It does not collect images. Examination of our Use Case reveals that we have an extension where new images are collected, however we don't have an actor to perform that task.

- Drag the **Actors** icon from the Diagram Palette onto the **Collect and Deliver New Product** element.
- Select **Collectors**, click add and click **OK**.

The resulting Use Case Diagram is shown below.





# 5

## Building the Behavior Model

- In this section, you define the behavior of your system
- Define the Top-Level Behavior
  - Refine the Behavior
  - Utilize Various Diagram Views
  - Establish Requirements Traceability

## Building a Functional Model

We will do this graphically by building either an Activity Diagram or a Functional Flow Block Diagram (FFBD) of our context function, **\_Perform Physical Context Functions**. (We will use the Activity Diagram.) CORE's Activity Diagrams/EFFBDs have the classic features of logic structures and functional decomposition. The logic constructs allow you to indicate the control structure and sequencing relationships of functions.

### Creating Function Elements

When the sample project was created, the Model Assistant was enabled. The Auto-create Root Function option allows CORE to automatically create a new **Function** element whenever a new **Component** is created. It also creates a performs/allocated to relationship between the two along with the relation attribute **Behavior Type** set to **Integrated (Root)**. CORE names the new **Function** **\_Perform Component name Functions**.

(NOTE: the Model Assistant also auto creates a root State for new Components.)

Let's view those **functions** and number them in Table View.

- Switch to Table View using the *Table View* icon in the Toolbar.
- Select the **Function** class in the Project Pane.
- We need to rename **\_Perform Debris 3 Functions**. Remember Debris 3 was the **component** we transformed from a **requirement**. Change the name to **\_Perform Geospatial Library Functions**.
- Use the table below to number the **Functions**.

	Number	Name
1	0	_Perform Geospatial Library Functions
2	C	_Perform Physical Context Functions
3	C.1	_Perform Customers Functions
4	C.2	_Perform Certification Authority Functions
5	C.3	Perform Collectors Functions

- Switch to *Element List* view.
- In the Element Browser, select the **Function** element **\_Perform Physical Context Functions**.
- Click the *Activity Diagram* tab  **Activity** to open an Activity Diagram of the selected element.

**NOTE**  
CORE Essentials users can complete the steps in this section in the EFFBD.

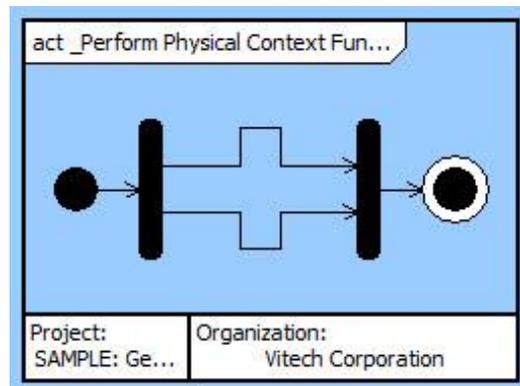
### Inserting a Parallel Structure

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

- **\_Perform Geospatial Library Functions**,
- **\_Perform Customer Functions**,
- **\_Perform Collector Functions**, and
- **\_Perform Certification Authority Functions**.

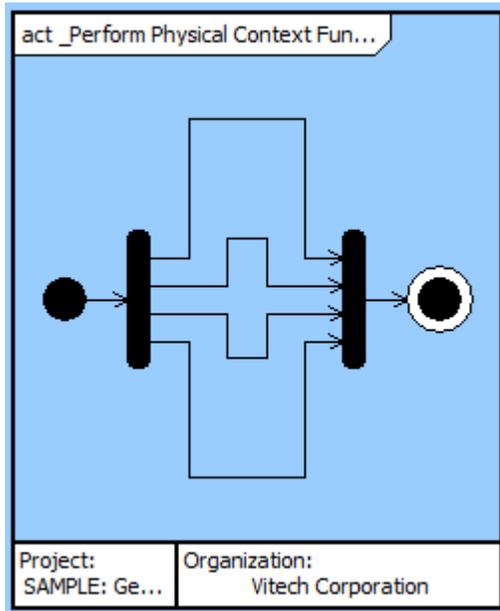
Each of these functions works in parallel. To incorporate these functions into our diagram, we will create parallel branches.

- From the Diagram Palette, click on the **Parallel** icon  **Parallel** and drag-drop on top of the main branch.





- To add more branches select the *Add Branch* icon  and drag drop it on top of either end of the AND construct.
- Repeat the drag-drop of the *Add Branch* icon so that you have four branches as shown below.



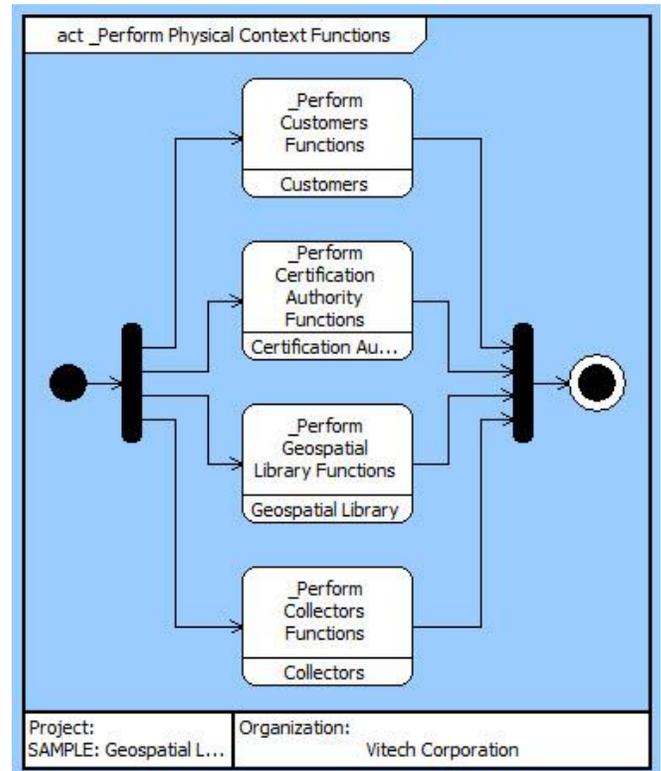
## Adding Functions to an Activity Diagram

Now, using drag-drop, we'll insert one **function** on each branch of the diagram.

The Key Entities tab shows the elements that can be inserted. You can select from existing elements or create new ones.

- From the Diagram Palette, select **Key Entities**.
- Select the **Function** class.
- Drag **\_Perform Customers Functions** onto the top branch.
- Repeat this step to add **\_Perform Certification Authority Functions** on the second branch, **\_Perform Geospatial Library Functions** on the third

branch, and **\_Perform Collectors Functions** on the



fourth (bottom) branch.

Compare your diagram to the one shown on the right.

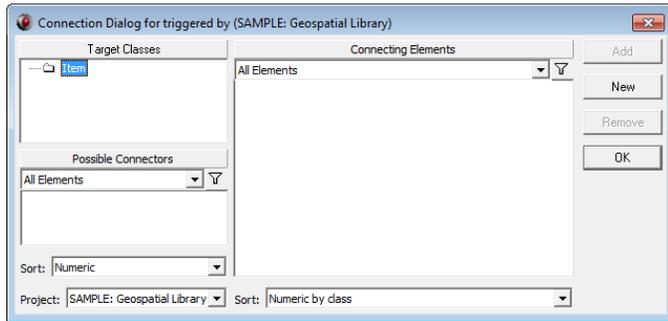
## Adding Inputs and Outputs

Now let's add inputs and outputs to these root functions to show the interdependencies of the **Functions**. While we are doing this example using the Activity Diagram, the process can be accomplished using the EFFBD and N2 Diagrams.

Input and outputs are represented in CORE as **Items**. Items may be connected as either data-stores or triggers. In a trigger-type connection, the receiving function will not execute until the "trigger" **Item** has 'arrived.' With data-stores type connections, the function will be executed independent of the item's 'arrival.'

- In the Activity Diagram, select **\_Perform Customers Functions**.
- Hold the *Shift* key down and select **\_Perform Geospatial Library Functions**.

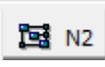
- Click the *Connect Via Trigger* icon  on the diagram toolbar to open a Connection Dialog.



- Double click on the **Item** folder to create a new element.
- Name the new element **requests**.
- Click **OK**.

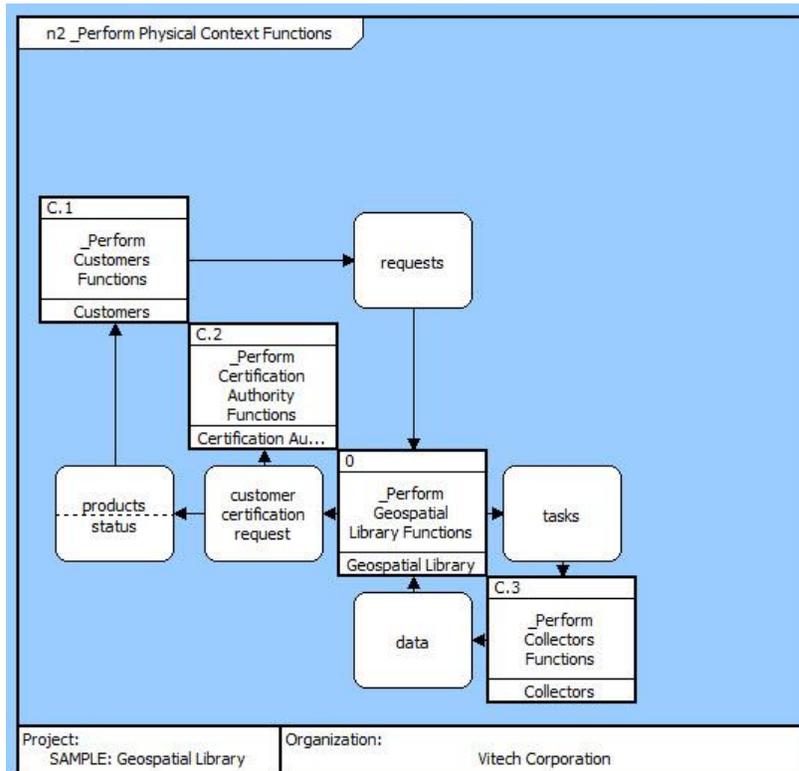
**Note**  
When using the Shift-Click method to select elements to connect, be careful about the order in which you click select them. The first element selected will be the sending function. The second element selected will be the receiving function.

To show the variety of ways you can add and represent data in the system design repository, let's switch to the N2 diagram to finish adding inputs and outputs.

- Click the *N2 Diagram* tab .
- Using the table below as your guide, complete the model using the same Shift-Click method. Add the following additional items: **products**, **status**, **customer certification request**, **tasks**, and **data**.

Item	Output from	Triggers
products	_Perform Geospatial Library Functions	_Perform Customers Functions
status	_Perform Geospatial Library Functions	_Perform Customers Functions
customer certification request	_Perform Geospatial Library Functions	_Perform Certification Authority Functions
tasks	_Perform Geospatial Library Functions	_Perform Collectors Functions
data	_Perform Collectors Functions	_Perform Geospatial Library Functions

The resulting diagram should be similar to the following diagram:



## Deriving the Behavior for Our System

Now that we have defined the context for our system, we will now decompose the root **function** for our system `_Perform Geospatial Library Functions`.

Because this Guided Tour is intended as an introduction to CORE we will not work through the entirety of the system. The Geospatial Library model is simply too complex to include all details in this Guided Tour.

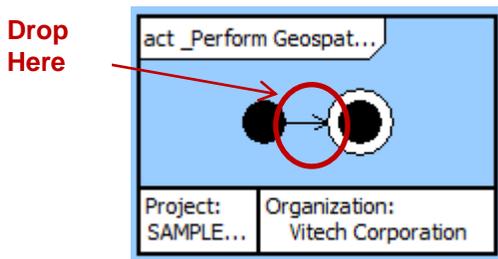
We'll focus instead on a very basic decomposition. If you would like to examine the full decomposition and system behavior, you can do so by importing Geospatial LibrarySampleSolution.xml found in the `ICORE 9\Data\Samples` folder. (Importing projects was addressed earlier in Section 2 of the Guided Tour.)

This work can be completed in either the Enhanced FFBD (EFFBD) or the Activity Diagram. We will work in the Activity Diagram.

- Select the `_Perform Geospatial Library Functions` element in the Elements pane.
- Click the *Activity Diagram* icon  in the Toolbar to open the view in a new window.

We'll use the Diagram Palette exclusively during this process. Keep in mind that while we are building our diagram visually, a data model is actually being built for us in the background by CORE and all diagrams are merely visual representations of the underlying model.

- Drag the *New Node* icon  `New Node` onto the diagram and drop it on the branch between the two end nodes.



The *New Node* command names the element with a generic name. We need to be much more specific.

- Select the element then right-click on `Function_001` and select *Rename Element*.
- Name the element **Accept Request**.

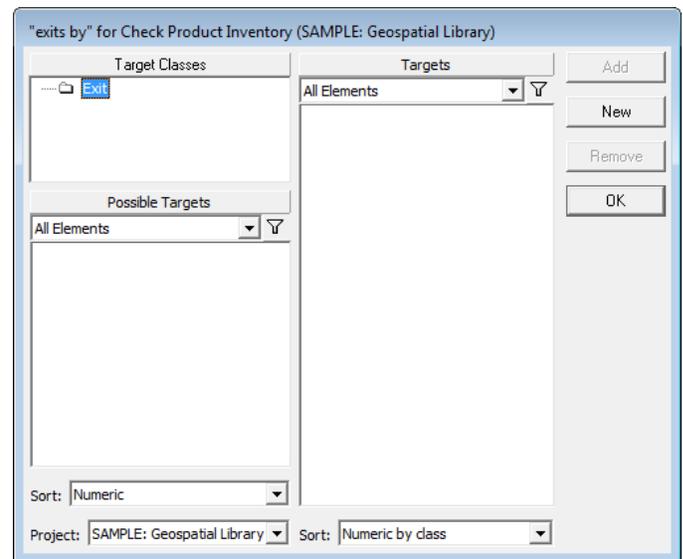
### Note

Remember that this two-step process can be accomplished in one step by holding the **Ctrl** key while dropping the element, this shortcut automatically opens the rename element dialog box.

- Drop another *New Node* construct onto the branch **AFTER** the Accept Request element.
- Rename this second element **Check Product Inventory**.

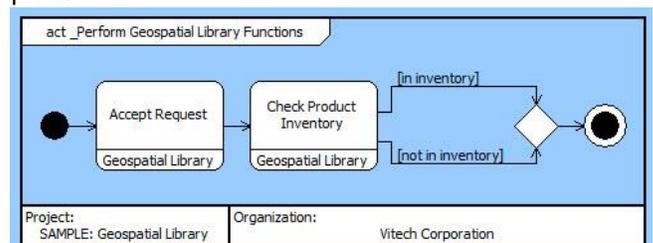
The outcome of the Check Product Inventory function has two possibilities: **in inventory** and **not in inventory**. We'll capture that with multiple exit conditions.

- From the Palette, drop the *Exit Condition* icon  onto the **Check Product Inventory function**.



- In this dialog, create two new elements: **in inventory** and **not in inventory**.
- Click **OK**.

Below is the Activity Diagram with this multi-exit function in place.



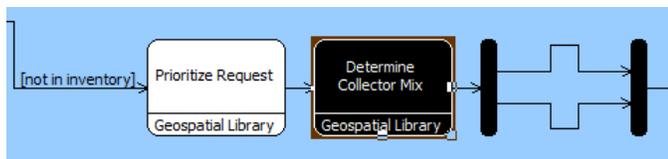


The “not in inventory” branch will be our main area of focus.

- Drop a *New Node* construct onto the **not in inventory** branch.
- Rename this element **Prioritize Request**.
- Add another node AFTER **Prioritize Request** named **Determine Collector Mix**.

While the system is getting the image it will also notify the end user of the estimated delivery schedule. We’ll use a a Parallel construct to represent this.

- Drop a *Parallel* construct onto the end of the **not in inventory** branch.



- On the top branch of the Parallel construct drop a *New Node* and rename it **Notify User of Estimated Schedule**.
- On the bottom branch drop a *New Node* and rename it **Task Collectors**.
- Now working at the end of the branch (after the Parallel construct), add two more elements in this order:
  - **Accept and Format Collector Products**
  - **Put Product In Inventory**.

The Accept and Format Collector Products icon may show up with truncated text in the ellipse (depending on your screen size and resolution). If this happens the following steps can be taken to size the icon so that all text is viewable.

- Click in the background of the diagram to ensure that nothing is selected.
- Click the *Auto-size* icon  in the Diagram Toolbar, then click Auto-Size icons.
- Click *Yes* to auto-size all diagram content.

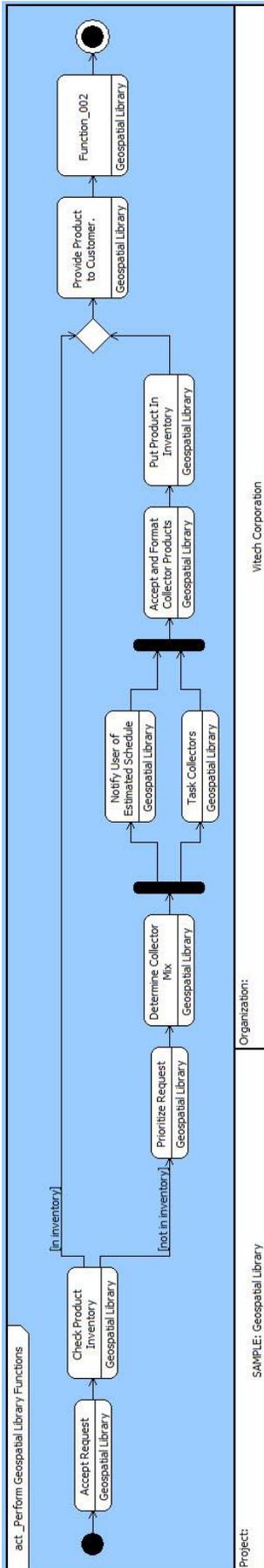
#### Note

This command can also be used to auto-size individual nodes. Select the desired node, then click the *Auto-size* icon.

We’re done with the “not in inventory” branch and are ready to add two final elements to the end of the basic behavior of our system.

- Place two more *New Node* icons at the end of the primary branch.
- Name these elements **Get Product From Inventory** and **Provide Product to Customer**, in that order.

Compare your diagram to the one on the next page.



## Adding Inputs and Outputs in an Activity Diagram

Inputs, outputs, and triggers can be added directly to the **function** in many of the functional diagram representations. For the following steps we will continue to work in an Activity Diagram. However, you can work in another behavioral diagram if you prefer.

Just as we added triggers in the behavior diagram of our Context function, we'll do the same for the Geospatial Library behavior.

Note the difference between triggers and inputs. If a function has a trigger, then the trigger is required to be present before the function can start. Inputs are not required for the function to start.

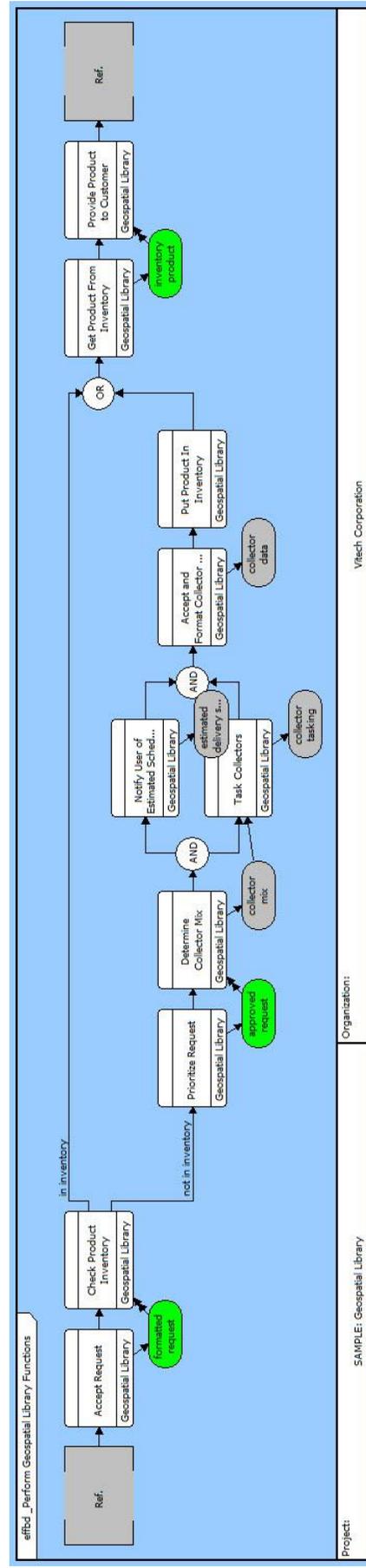
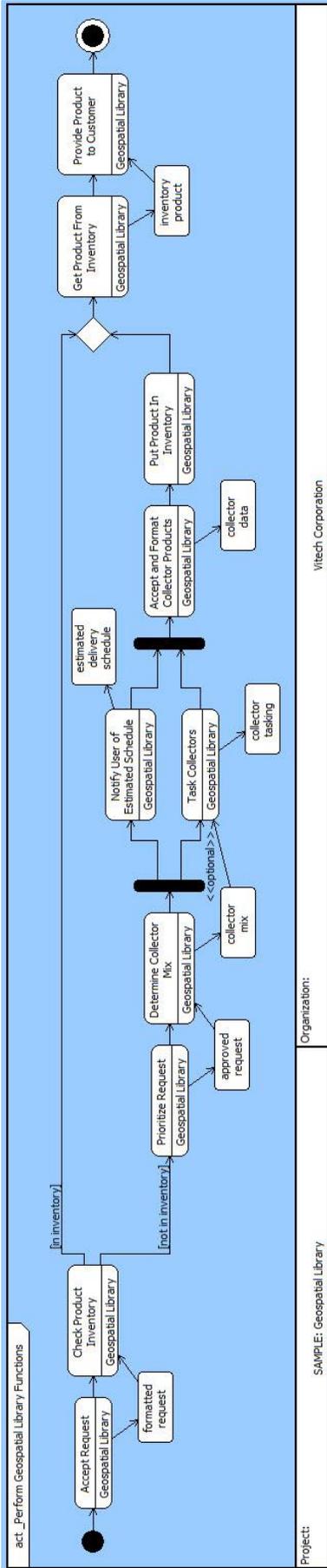
- Add **items** to the behavior model using the table below. None of these **items** currently exist so you will need to create them as you go.

Items	Actions on Functions		
	<u>output from</u>	<u>Inputs to</u>	<u>triggers</u>
formatted request	Accept Request		Check Product Inventory
collector mix	Determine Collector Mix	Task Collectors	
approved request	Prioritize Request		Determine Collector Mix
estimated delivery schedule	Notify User of Estimated Schedule		
collector tasking	Task Collectors		
collector data			Accept and Format Collector Products
inventory product	Get Product From Inventory		Provide Product to Customer

After you have added your **items** to the behavior view, take a moment to adjust the diagram for better layout and presentation.

- Use the *Auto-size* icon to expand icons.
- Click and drag **items** to reposition them if desired. Be careful to always drop them on the background. If you drop them on functions, CORE will create relationships between the **item** and **function**.

The next page shows both the Activity Diagram and EFFBD of the Geospatial Library element. Compare your diagram to ensure completeness.



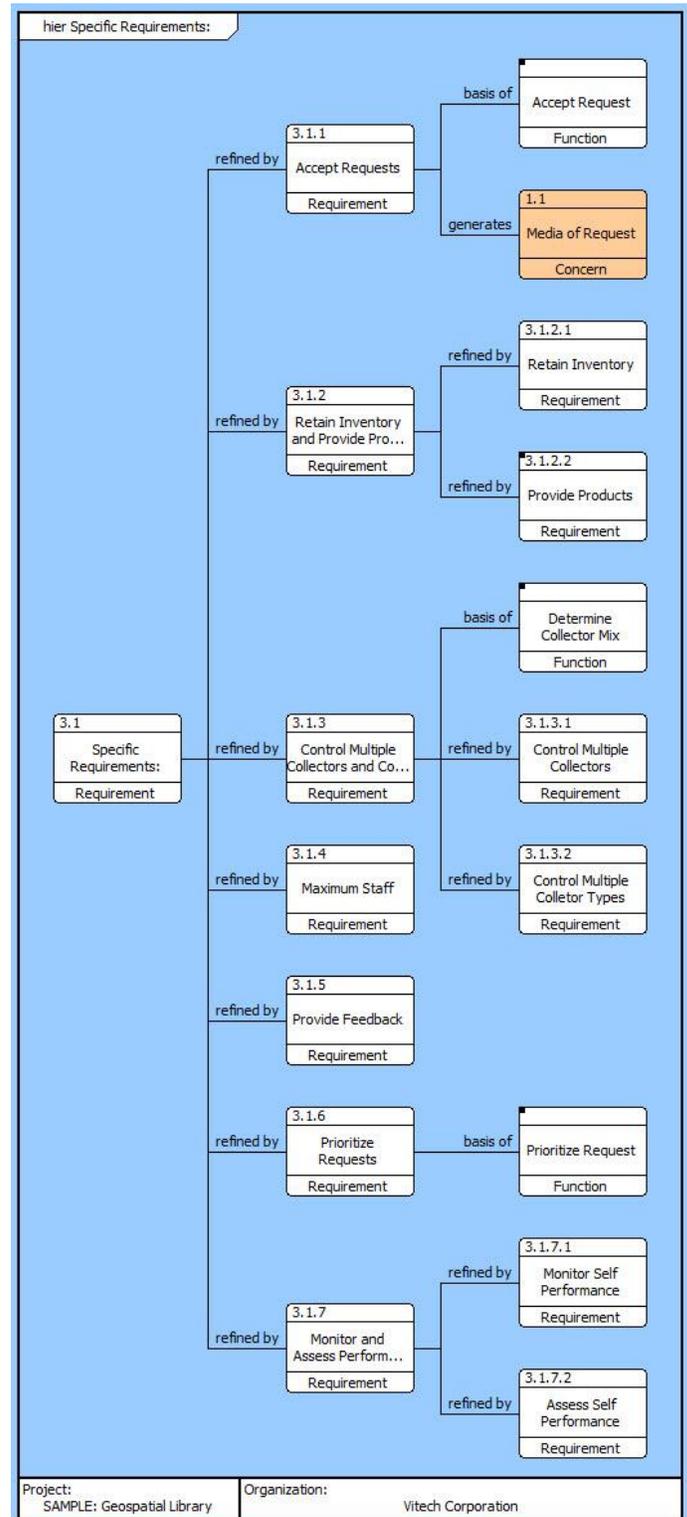
## Adding to the Traceability

Now that we have defined our system and its behavior, let's extend the requirements traceability to identify the elements that fulfill each **requirement**. We will use the Element Browser to add these traceability relationships.

- In the Element Browser, select the class **Requirement** and then the element **3.1.2.2 Provide Products**. Double-click the basis of relationship to open an Edit Targets dialog.
- In the Edit Targets dialog, select target class **Function**. Select possible target **Provide Product to Customer**. Then click *Add*. Click *OK*.
- Use this process to complete the remainder of the basis of relationships shown in the diagram to the right.

### NOTE

A black dot in the upper-right hand corner of an icon means that icon is repeated somewhere else in the diagram. To see where else it appears, right-click on the icon of interest, and from the drop-down menu, select the *Highlight Matching Nodes* command in the pop-up menu.



THIS PAGE INTENTIONALLY BLANK

# 6

## Completing the Physical Model

In this section, you complete the physical model

- Create Components Below the System
- Allocate Functions to Components
- Create Physical Interfaces
- Assess the Model

### Extending the Component (Physical) Hierarchy

Our next step is to determine how to decompose the Geospatial Library **component**.

Let us now assume that our Geospatial Library is built from two **components**: **Workstation** and **Command Center**.

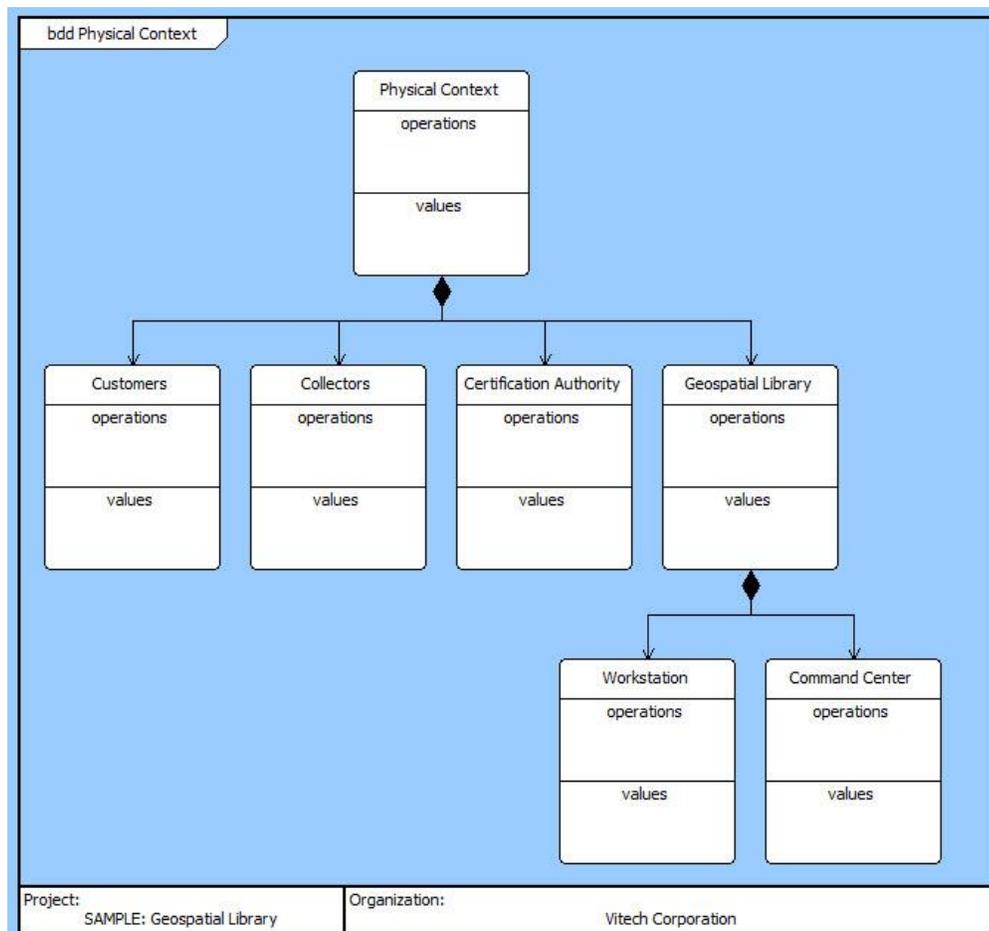
- In the Element Browser, create the two new **component** elements, **Workstation** and **Command Center**.

- Assign their respective **Number**, and **Type** attributes as shown below.
- For each component, establish the built in relationship with **SYS.1 Geospatial Library** as the target. Note that built in is the inverse of built from relationship.
- Now, open a Structure BDD of the **Physical Context component**. Do this by selecting the element **Physical Context**. Click on the *Structure BDD* icon  *Structure BDD*.

Compare your diagram to the one below.

#### NOTE

CORE Essentials users can complete the steps on this page in the Physical Block Diagram. Drop the New Node icon on the diagram background instead of on an element.





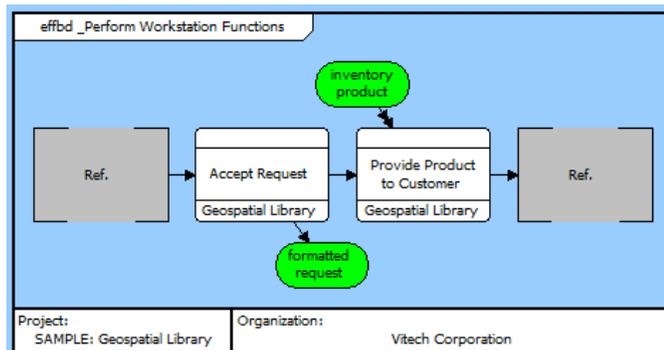
## Allocating the Functions

Now that we have completed both our behavior model and our component hierarchy, we will allocate the leaf-level **Functions** to the **Components** that will perform them. The following EFFBD's illustrate how the GeospatialLibrary behavior was partitioned to the Workstation and Command Center.

Notice that the Auto-allocation on Decomposition feature of the Model Assistant has automatically created root **functions** for our new **Components** and established allocated to/performs relationships to the **components**.

- In the **Function** class, select the **\_Perform Workstation Functions** element.
- On the Property Sheet of the element, double-click the decomposed by relationship to open an edit targets dialog.
- Add two functions: **Accept Request** and **Provide Products to Customer**.

Compare your EFFBD to the one shown below.



Creating a decomposed by relationship will place the elements in the behavioral view of the parent as shown in the diagram above.

- Select the **\_Perform Command Center Functions** element.
- On the Property Sheet of the element, double-click the decomposed by relationship to open an Edit Targets dialog.
- Add the following functions:
  - **Accept and Format Collector Products**
  - **Check Product Inventory**
  - **Determine Collector Mix**
  - **Notify User of Estimated Schedule**
  - **Prioritize Request**
  - **Put Product In Inventory**

### Task Collectors

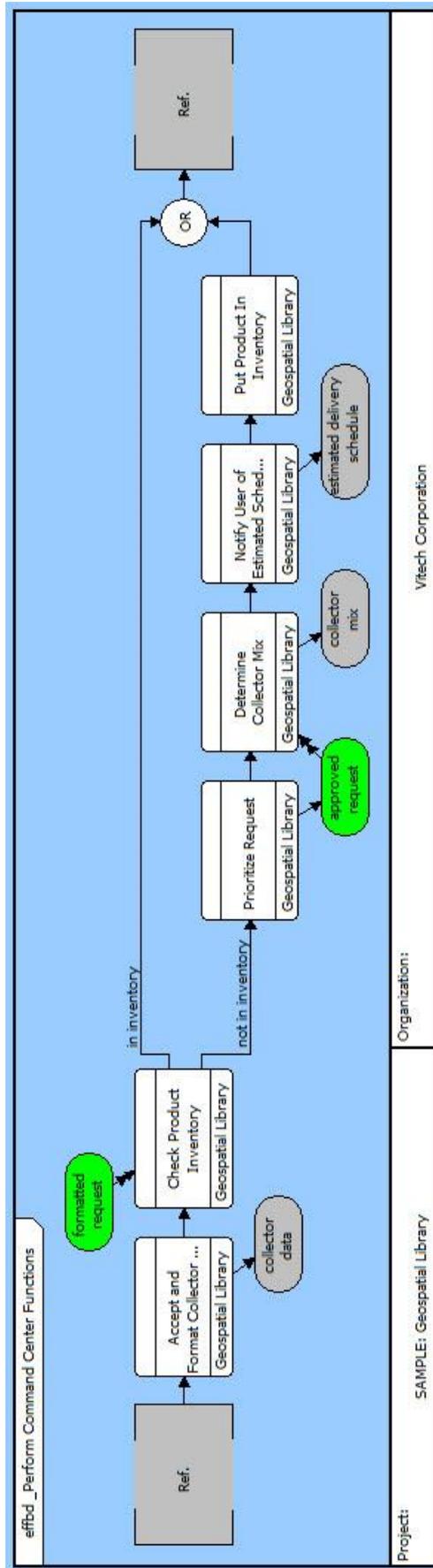
- View the EFFBD of **\_Perform Command Center Functions**.

Just as with the previous set, CORE placed these elements on the behavior view of the parent. It did not create the needed logic. We need to move several elements onto the bottom branch of the OR construct as well as to reorder them.

- Select **Prioritize Request**.
- Press **Ctrl + X**.
- Click on the **not in inventory** branch.
- Press **Ctrl + V**.
- Repeat this process to move and reorder the functions to match the diagram on the next page.

#### Note:

You can also move elements by holding down Ctrl and selecting the element. As soon as you start moving the element, you can release the Ctrl key. Drop the element in the desired location.



## Completing the Physical Model

By allocating our leaf-level **Functions** to **Components**, we have established logical interfaces between our **Components**. We will formalize this in our model by establishing physical connections between our **Components**.

The **Link** Class is used to represent the physical connection between components.

- In the Project Explorer select the **Component Physical Context**.
- Open a Flow Internal Block Diagram (Flow IBD) .
- Select the **Geospatial Library** element.
- Shift-click to select **Customers**.
- Click the **Connect** icon  on the Diagram Toolbar.
- In the Connection Dialog, create three new elements: **Return Link**, **Status Link**, and **Request Link**.
- Click **OK**.

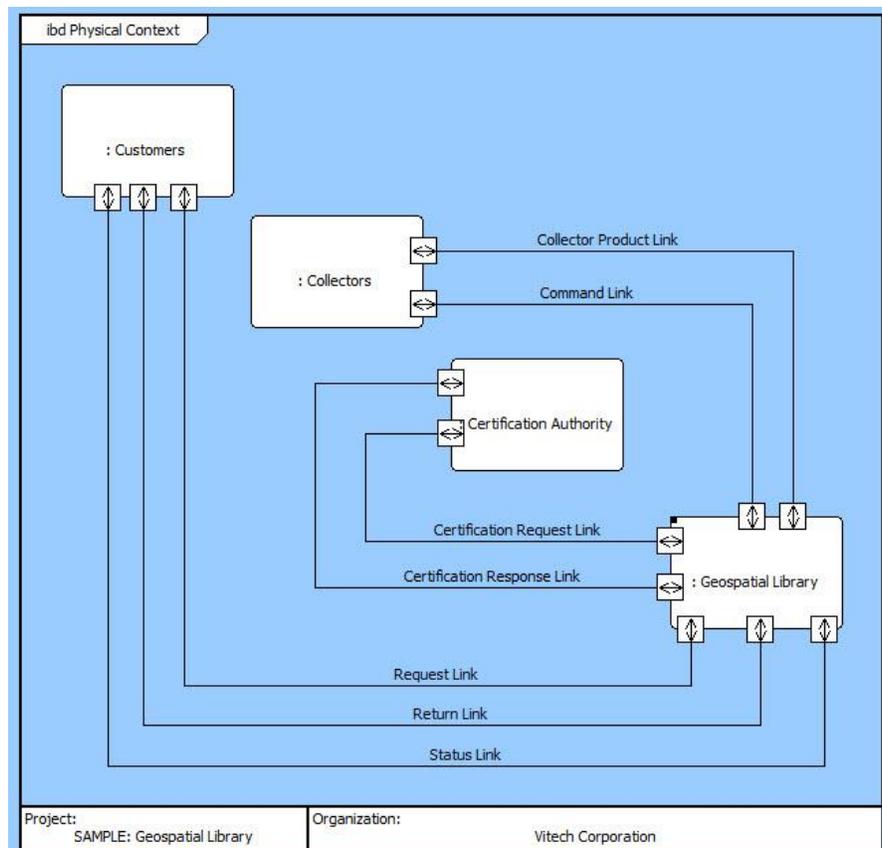
- Repeat the process to add more connections as defined in the diagram below.

### Note:

The default layout for the links may need to be adjusted. To do this, select the link you want to adjust and handles will appear. Drag the handles to adjust the link placement.

### NOTE

CORE Essentials users can complete the steps on this page in the Physical N2. The layout is different, but the steps are the same.



Now that we've established our **Links**, we need to define what they transfer. We identified **items** that are transferred when building our behavior model and we will use those same **items** here.

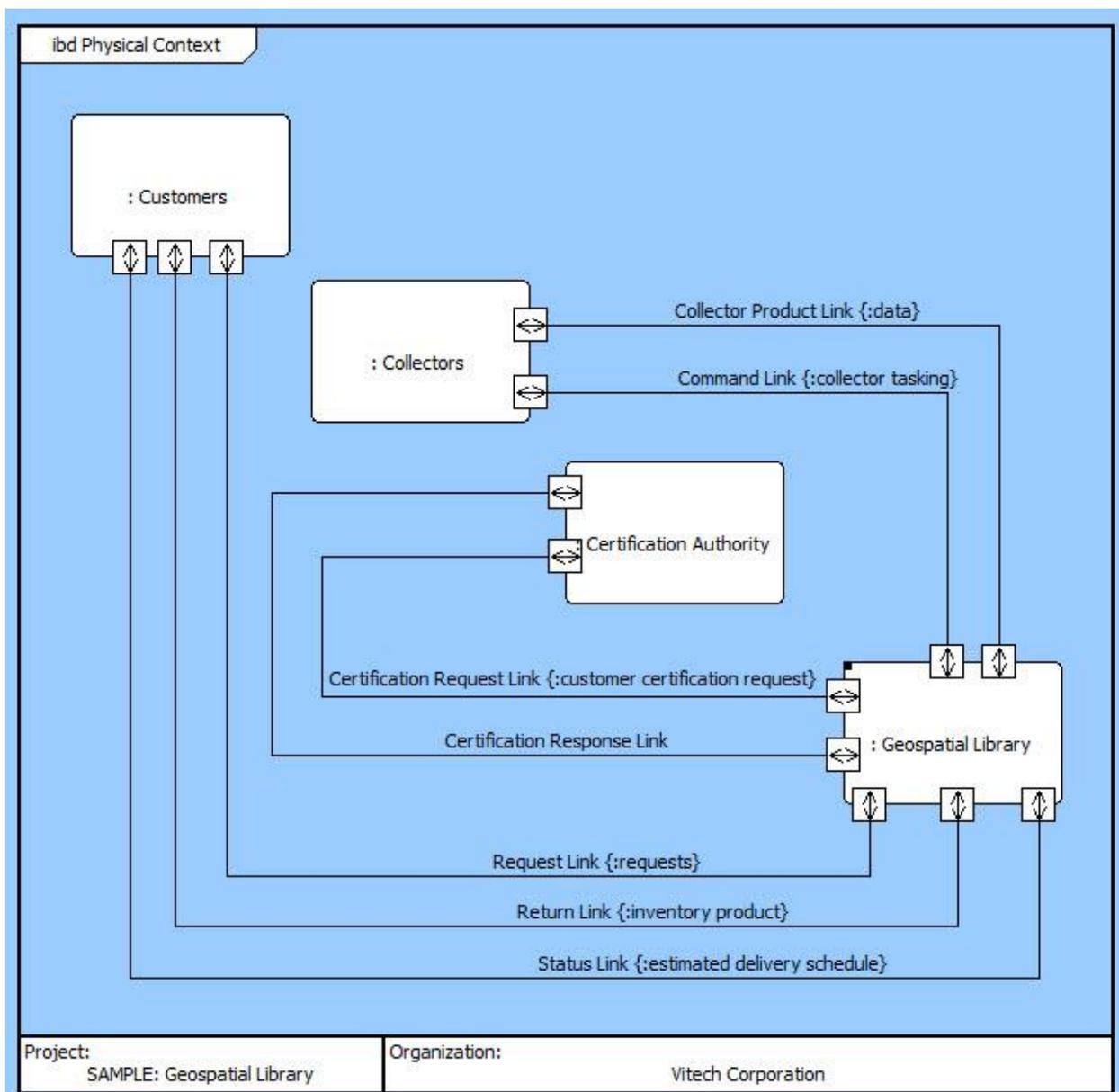
- In the Diagram Palette, select the *All Entities* tab.
- Select the **Item** class to display the list of items.

We can now drag and drop the needed **items** right onto the **links** to establish that the links transfer those items. Each time you drag and drop an **Item** onto a **Link**, the Relation Dialog Box will open to confirm the relation you are making.

- Use the diagram below as a reference to to make additional relationships between **Links** and **Items**.

#### Note

You may need to adjust the layout of the diagram to improve the readability of the diagram. **Components** can be resized by dragging the edges of the element to the desired size. **Links** can be repositioned by first clicking on the line, then dragging the handle on the end to the desired location.





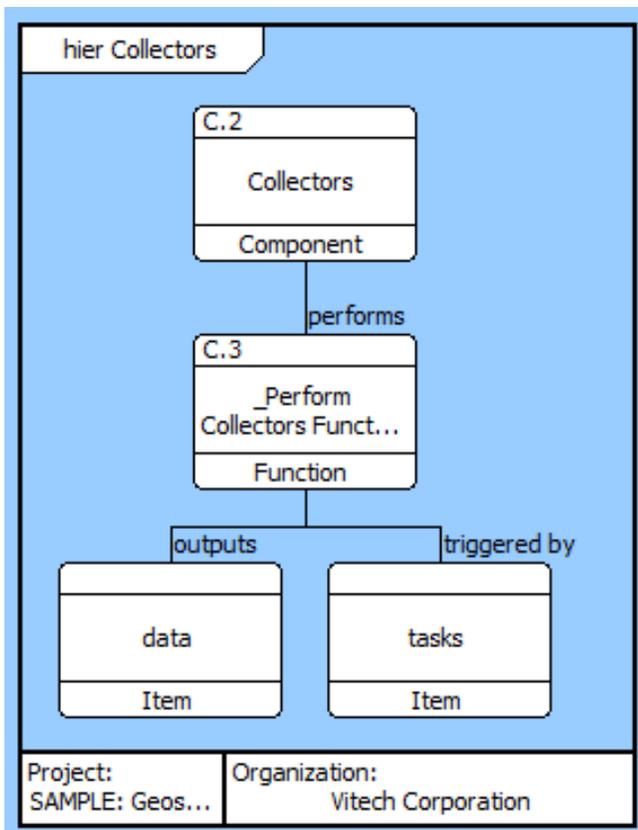
## Impact Analysis

To show the power of developing our system model with diagrams and seamlessly updating the integrated system design repository, we will walk through a typical systems engineering analysis using system diagrams.

Suppose that the customer wants to know the impact of exchanging out/replacing the Collectors.

- Select the **component Collectors** in the Project Explorer.
- Click either the *Spider Diagram*  or *Hierarchy Diagram* icon .
- Right-click on the diagram, select Hierarchy Type and select *Behavior Impact of Physical Change* from the list of stored definitions and click *OK*.

A diagram similar to the one below will open. This diagram allows us to visually identify all the functions and items associated with the selected component.



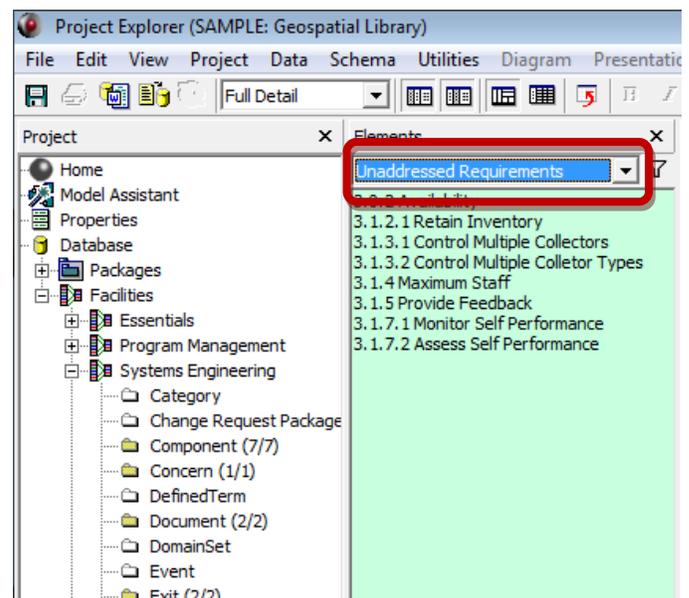
## Ensuring Full Traceability from Source Document to Physical Architecture

Traceability indicates which parts of a system design satisfy specific source requirements and allows easy impact assessment of changes to system requirements. Reverse traceability provides the ability to determine specific components which have been defined without benefit of system analysis. Traceability also supports internal consistency, which is difficult to attain when tracing elements from document to document without the benefit of an integrated design repository. This means that all incompatibilities are checked and resolved, the design is complete (all interfaces and environments are specified), and the design is feasible (all critical elements are demonstrated).

Using relationships for traceability makes it easy to detect unfulfilled requirements and unresolved concerns.

- Close the Hierarchy Diagram window.
- Select the **Requirements** class in the Project pane.
- At the top of the Elements pane, select *Unaddressed Requirements* from the filter drop-down list. This will filter the elements list to display only those **Requirements** that do not have targets for the basis of, refined by, and specifies relationships.

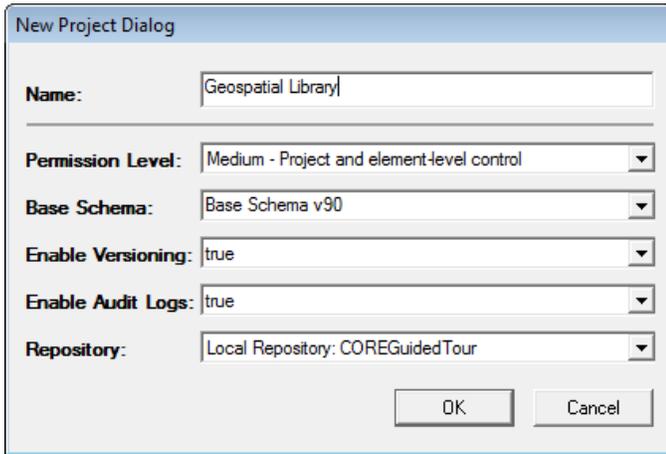
For example, we see that **requirement 3.1.4 Maximum Staff** has not yet been addressed. A Traceability Hierarchy Diagram does not have to be opened to determine that 3.1.4 has not yet been addressed.



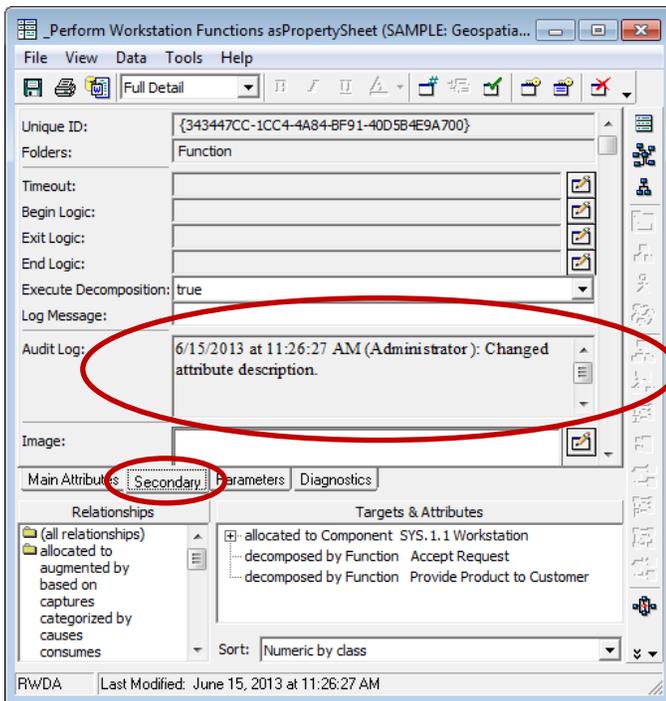
## Change Control

CORE provides several mechanisms for identifying and tracking changes to the repository. These include Audit Logs, Versioning, and a Project Compare Report.

Audits Logs provide a history of changes to an element. When a new project is created Audit Logs are enabled by default.



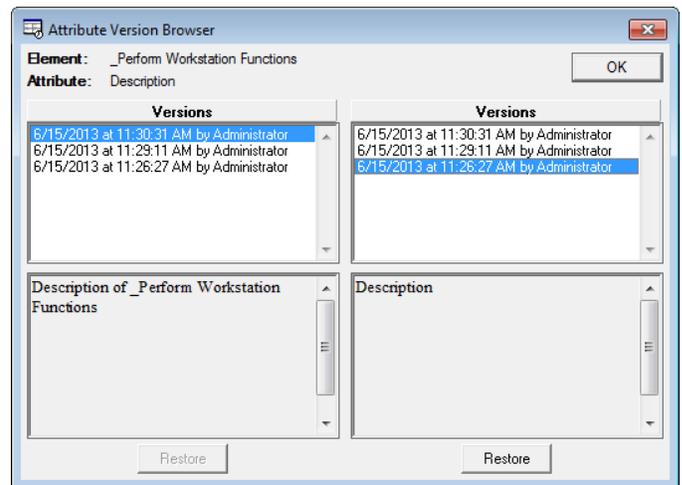
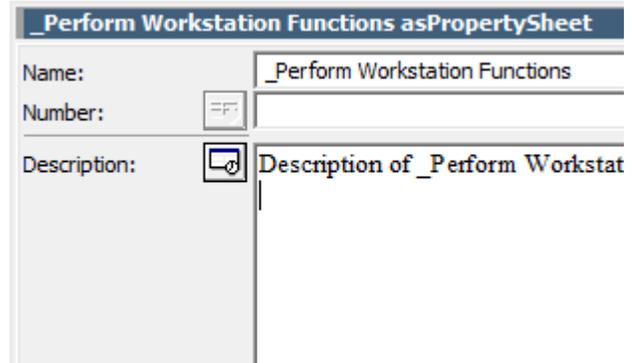
The Audit Log is accessed through a Property Sheet's Secondary tab.



The log identifies when the change was made, by whom, and the nature of the change.

Versioning provides a more detailed view of changes to an element's attributes. When an attribute is changed the

Version Browser  icon becomes highlighted. Select the icon to open a Versioning dialog.



The dialog identifies when the change was made, by whom, and the attribute's value. The bottom value is the element's baseline. The top value is the element's current value. Pressing the **Restore** button make the highlighted value the current value with its original timestamp.

In the Project Explorer **Project** menu you'll find the commands to **Purge Versions** and **Baseline Versions**. Purge Versions removes all interim values leaving the current and baseline values. Baseline Versions removes all values except the current, which become the baseline. These apply to the entire project. Selective purging and baselining can be done using the reports Purge Attribute History and Baseline Elements. The Attribute History Report prints the version history for selected elements and attributes.

There is another report known as Project Compare. This report is a comprehensive comparison of two projects. It lists new and deleted elements, changes to attributes and relations, and identifies when a diagram has been modified.

THIS PAGE INTENTIONALLY BLANK

# 7 Generating Documentation

In this section, you will generate output from CORE

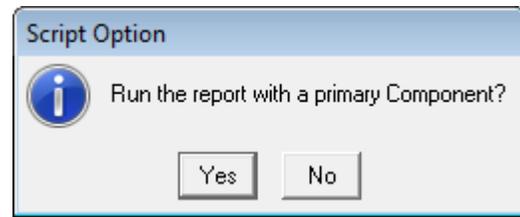
- Output a System Description Document
- Generate TeamView
- Review Standard Reports Available in CORE

## Generating a System Description Document (SDD)

A System Description Document (SDD) presents your project's primary systems engineering elements in a structured manner for review of your physical and behavior model and related information. For each element appearing in the SDD, key attributes and relationships are listed. User-selected diagram types are also included, as appropriate.

Typically, an SDD is generated for the system or one of its lower-level components. Only the elements directly or indirectly related to the selected component and its physical hierarchy are included. Alternatively, all systems engineering elements can be included.

- In the Project Explorer, click the *Run Script* icon  on the toolbar or select *Tools > Run Script* or enter *Ctrl+R* to open a Select Script dialog.
- In the Select Script dialog, you can use the Folder drop-down list to show **All Reports** or list a subset of reports by selecting a category. For now, choose **All Reports**. Then, use the *pull-down arrow* to select **System Description Document** from the Script drop-down list. (Reports are listed in alphabetic order on the All Reports menu.)
- Click *OK*.
- Answer *Yes* to run the report with a primary **Component**.

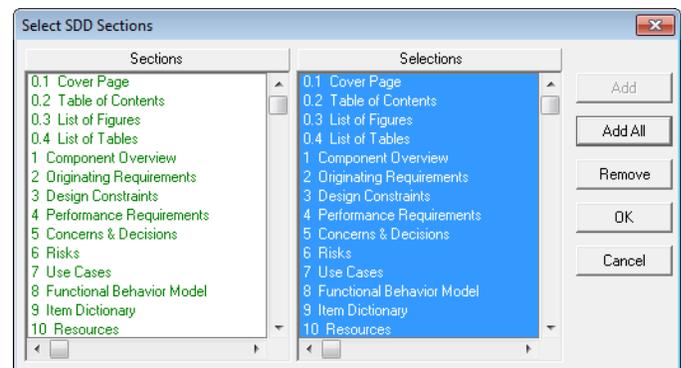


Next, you need to select the primary **Component** in the repository for the report.

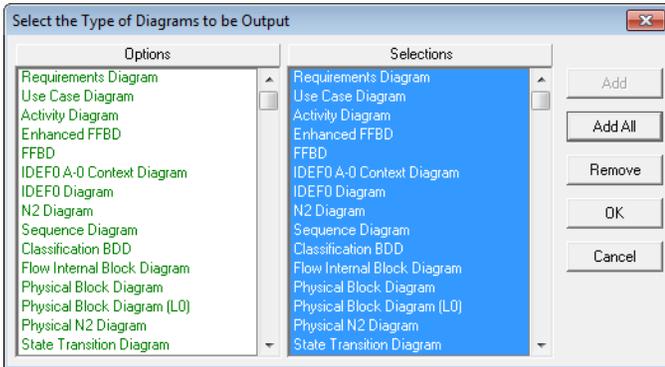
- Select **Geospatial Library** in the Candidates pane.
- Click *OK*.

When creating the System Description Document (SDD), you can select the sections of the SDD that you would like included. Select 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.

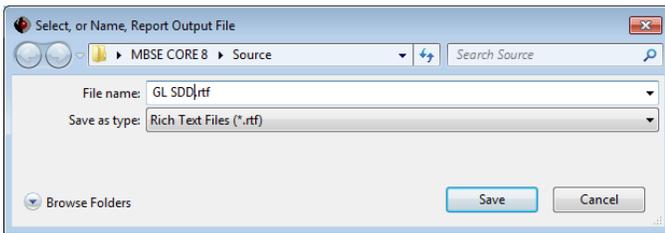
- Click *Add All* and then click *OK*.



- At the next prompt, select *Add All* to include all diagram types.
- Press *OK*.



- In the prompt for a report output file name, type **Geospatial Library SDD.rtf** and select the desired save location.
- Click *Save*.

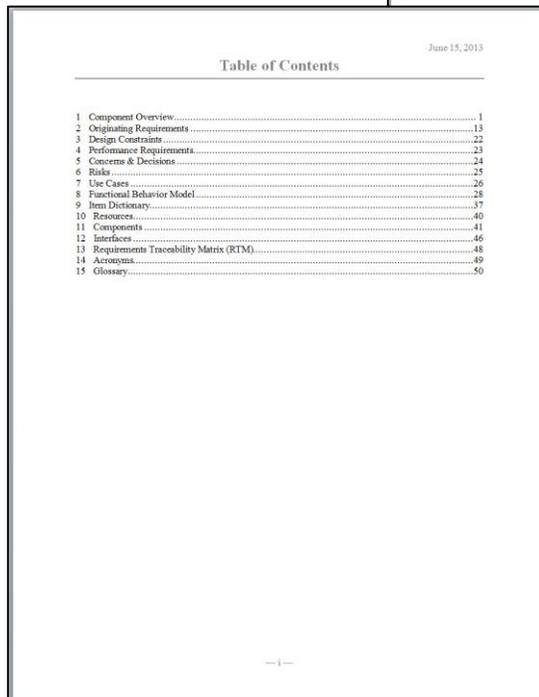
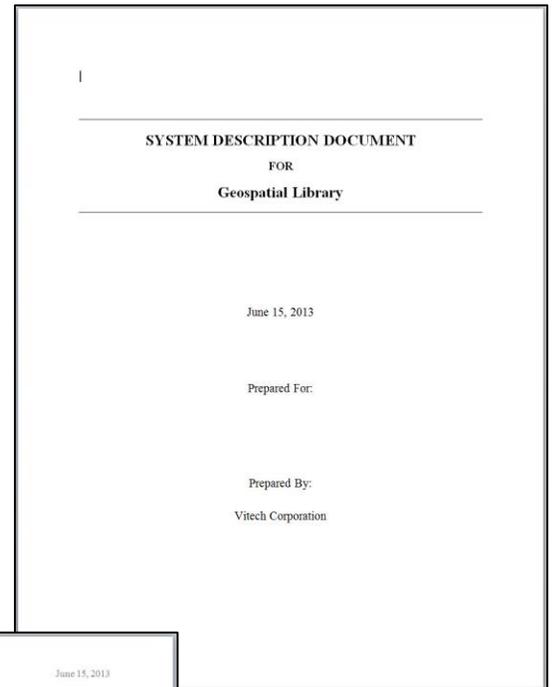


CORE generates the report as an RTF file. When completed, the report will automatically open in Microsoft Word for previewing and printing.

- When the script completes execution, you will be prompted with an Execution Completed dialog. Click *OK*.

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

- Type *CTRL + A* to select the entire document.
- Press the *F9* function key on your keyboard. Word will format and paginate the document for each table/list.



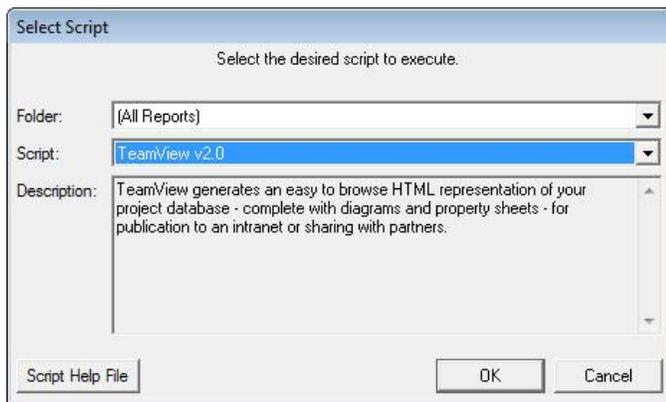
## Generating TeamView

Another report available in CORE outputs the contents of the system design repository in HTML format and generates a Homepage accessible by anyone with a Web browser. (This capability was formerly called the HTML Report.)

- To run this report, click the **Run Script** icon  on the toolbar.
- In the Select Script dialog, select **TeamView** from the Folder pull-down list. Select **TeamView 2.0** from the Script pull-down list. Click **OK**.

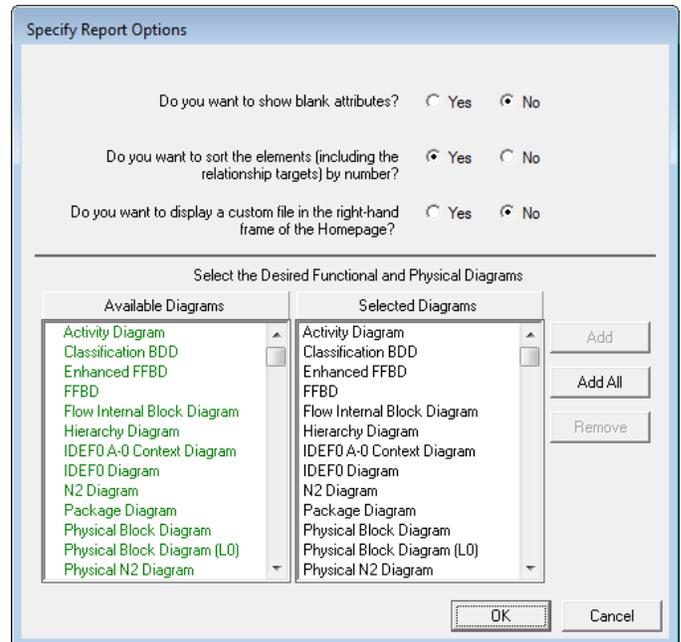
### User Tip

Notice that the Select Script dialog has a **Script Help File** button. Clicking this button will open a help file that explains what the script will output, the meaning of each prompt, and for complex outputs, where the information is drawn from the repository. This can help you confirm that you have linked your elements correctly.



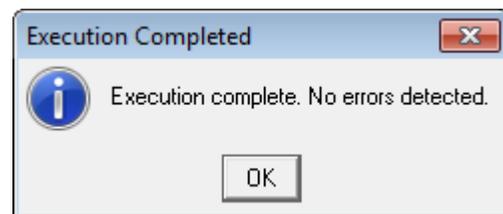
- When prompted to *Select a location for the Report Home Page* to be saved, create a folder named **SE Guided Tour** in the desired directory and click **Save** to accept the default filename (0\_homepage.htm) in the new directory.

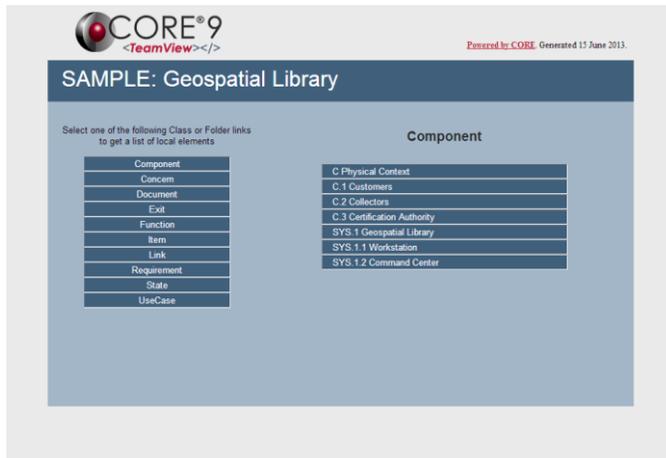
- When prompted to show blank attributes, click **No**.
- When prompted to sort the elements, click **Yes**.
- When prompted to select a custom file, click **No**.
- When prompted to select the diagram types, click **Add All**.
- Click **OK**.



When the report generation is complete, the Execution Complete window appears.

- Click **OK**.

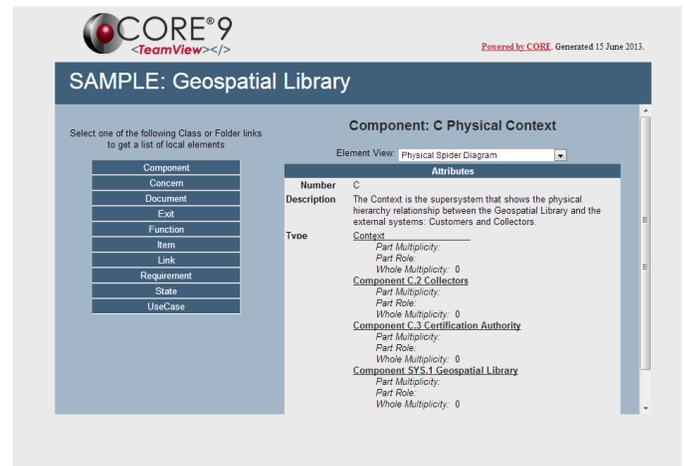




The file 0\_homepage.htm will open in your browser. Your page should look similar to the picture above.

Clicking on a class/folder link in the left-hand list displays the corresponding elements on the right. From this list, you can select any element to display its property sheet information.

- For example, click on the **Component** link. On the right, a list of all elements in the **Component** class in your project is displayed.
- Click on **C Physical Context** to show the elements information.

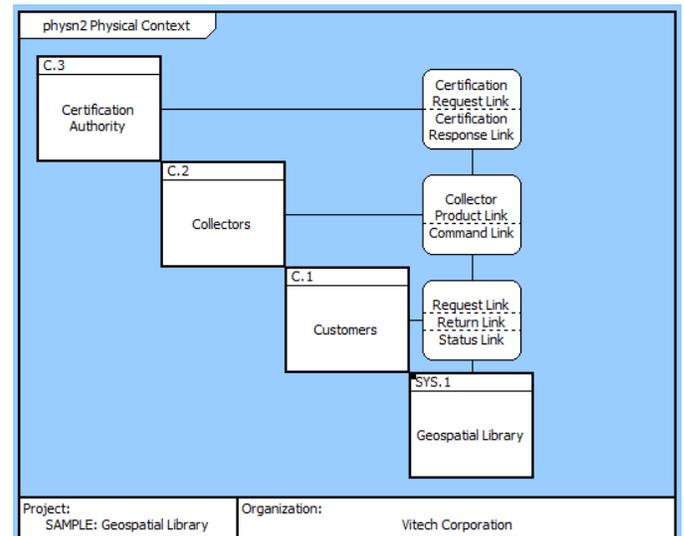


In the Relationships portion of the element information, each relationship Target is a link to the text view for that element. If the selected function has a diagram associated with it, that view is accessible by selecting the link from the drop-down menu located at the top of the text view as shown above.

- From the Element View drop-down menu, select **Physical N2 Diagram**. The Physical N2 of the component will be displayed in place of the element attributes and relationships. Each **component** and **link** icon is a link to that element.

**NOTE**

If you are using Microsoft Internet Explorer, you may get a warning message when you click on this function because it has an associated diagram file. Click **OK**. Then move your cursor to the security alert band in the browser and, from the pop-up menu, select **Allow Blocked Content**. When you get a final security warning, click **Yes**. You will then need to reselect **Function** and **GL Geospatial Library** to return to the element information. From here on, you will be able to view any diagram in the HTML output without a warning.



- Close the browser.

## Congratulations!

You have completed your first system design using CORE. Now, it's time to save your final model.

- Select *File > Save Repository As* from the Project Explorer toolbar to save your efforts.
- Selecting *Make Backup* causes Microsoft Windows to make a backup of your current repository file prior to saving the new repository file. This backup, named *repositoryname.bak*, can be renamed to a *.c90* file and used if necessary.
- Answer *Yes* to overwrite your previously saved repository.

With this guided tour as a desktop 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, you'll have more time for engineering your system.

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

## Standard Reports Provided with CORE 9

All of the scripts listed below are included with the full version of CORE 9.

### Common

- Print Formatted Text
- Set Up Rules

### Database

- Database Statistics Report
- Schema Definition Report

### EmbeddedConsistencyChecker

- Perform Consistency Checks – By Element
- Perform Consistency Checks – By Folder
- Perform Consistency Checks – By Package
- Perform Consistency Checks

### EmbeddedCustomChecker

- Perform Custom Checks – By Element
- Perform Custom Checks – By Folder
- Perform Custom Checks – By Package
- Perform Custom Checks

### Formal Documentation

- Create Document Subsections
- Delete Script Created Sections
- Duplicate Document Outline
- Generic Specification/Document
- Interface Requirements Specification (IRS)
- Save Document Templates
- Software Requirements Specification (SRS)
- System/Segment Design Documentation (SSDD)
- System/Segment Specification (SSS)
- Test & Evaluation Plan (TEP)

### PUID

- Assign Documentation PUIDs
- Clear Documentation PUIDs
- Update PUID registry

### Quick Reports

- Keyword Search
- Open Concerns Query

### Risk Report

- Risk Output
- Risk Rating
- Set Risk Rating

### Systems Engineering

- Attribute History Report
- Baseline Elements
- Element Definition
- Indented Hierarchy Report
- Keyword Search
- Open Concerns Query
- Project Compare Report
- Purge Attribute History
- Risk Output
- Risk Rating
- System Description Document (SDD)

### Utility

- Assign Documentation PUID
- Clear Documentation PUID
- Element from ID
- Structure Traversal
- Update PUID Registry
- V80 to v90 Project Migrator

### Versioning

- Attribute History Report
- Baseline Elements
- Purge Attribute History

### DoDAF

- AV-1 Overview and Summary Information
- AV-2 Integrated Dictionary
- CV-1 Vision
- CV-2 Capability Taxonomy
- CV-3 Capability Phasing
- CV-4 Capability Dependencies
- CV-5 Capability to Organizational Development Mapping
- CV-6 Capability to Operational Activities Mapping
- CV-7 Capability to Services Mapping
- DIV-1 Conceptual Data Model
- DIV-2 Logical Data Model
- DIV-3 Physical Data Model
- OV-1 High-Level Operational Concept Graphic
- OV-2 Operational Resource Flow Description
- OV-3 Operational Resource Flow Matrix
- OV-4 Organizational Relationships Chart
- OV-5a Operational Activity Decomposition Tree

- OV-5b Operational Activity Model
- OV-6a Operational Rules Model
- OV-6b State Transition Description
- OV-6c Event-Trace Description
- PV-1 Project Portfolio Relationships
- PV-2 Project Timelines
- PV-3 Project to Capability Mapping
- StdV-1 Standards Profile
- StdV-2 Standards Forecast
- SvcV-1 Services Context Description
- SvcV-2 Services Resource Flow Description
- SvcV-3a Systems-Services Matrix
- SvcV-3b Services-Services Matrix
- SvcV-4 Services Functionality Description
- SvcV-5 Operational Activity to Services Traceability Matrix
- SvcV-6 Services Resource Flow Matrix
- SvcV-7 Services Measures Matrix
- SvcV-8 Services Evolution Description
- SvcV-9 Services Technology & Skills Forecast
  
- SvcV-10a Services Rules Model
- SvcV-10b Services State Transition Description
- SvcV-10c Services Event-Trace Description
- SV-1 Systems Interface Description
- SV-2 Systems Resource Flow Description
- SV-3 Systems-Systems Matrix
- SV-4 Systems Functionality Description
- SV-5a Operational Activity to Systems Function Traceability Matrix
- SV-5b Operational Activity to Systems Traceability Matrix
- SV-6 Systems Resource Flow Matrix
- SV-7 Systems Measures Matrix
- SV-8 Systems Evolution Description
- SV-9 Systems Technology & Skills Forecast
- SV-10a Systems Rules Model
- SV-10b Systems State Transition Description
- SV-10c Systems Event-Trace Description

THIS PAGE INTENTIONALLY BLANK



# Appendix—Using CORE University Edition

This section summarizes the differences found in CORE University Edition

- Launch CORE University Edition
- Import a Data File into CORE University Edition
- Export (Save) a Data File from CORE University Edition
- Review Limitations and Available Reports

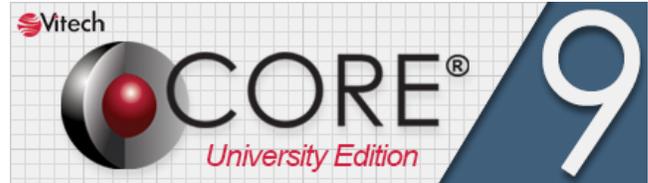
For information about our University Program, visit our website or contact our University Program Manager at [universityprogram@vitechcorp.com](mailto:universityprogram@vitechcorp.com). The University Edition of CORE is provided under special agreement for academic or evaluation use only. If you wish to use CORE for commercial or other purposes, please contact Vitech Corporation at (540) 951-3322 or via e-mail at [info@vitechcorp.com](mailto:info@vitechcorp.com). Vitech has an official price list, which provides for subscription and evaluation licenses for CORE as well as systems engineering training classes and purchased technical services.

## Opening CORE 9 University Edition

This section will show how to get started with CORE University Edition by opening CORE University, importing a sample file, and saving your work. Once CORE University is open and you have imported the sample data, you'll return to the Examining CORE section of this guided tour. However, certain features are inhibited and whenever these features are tried an appropriate message will appear.

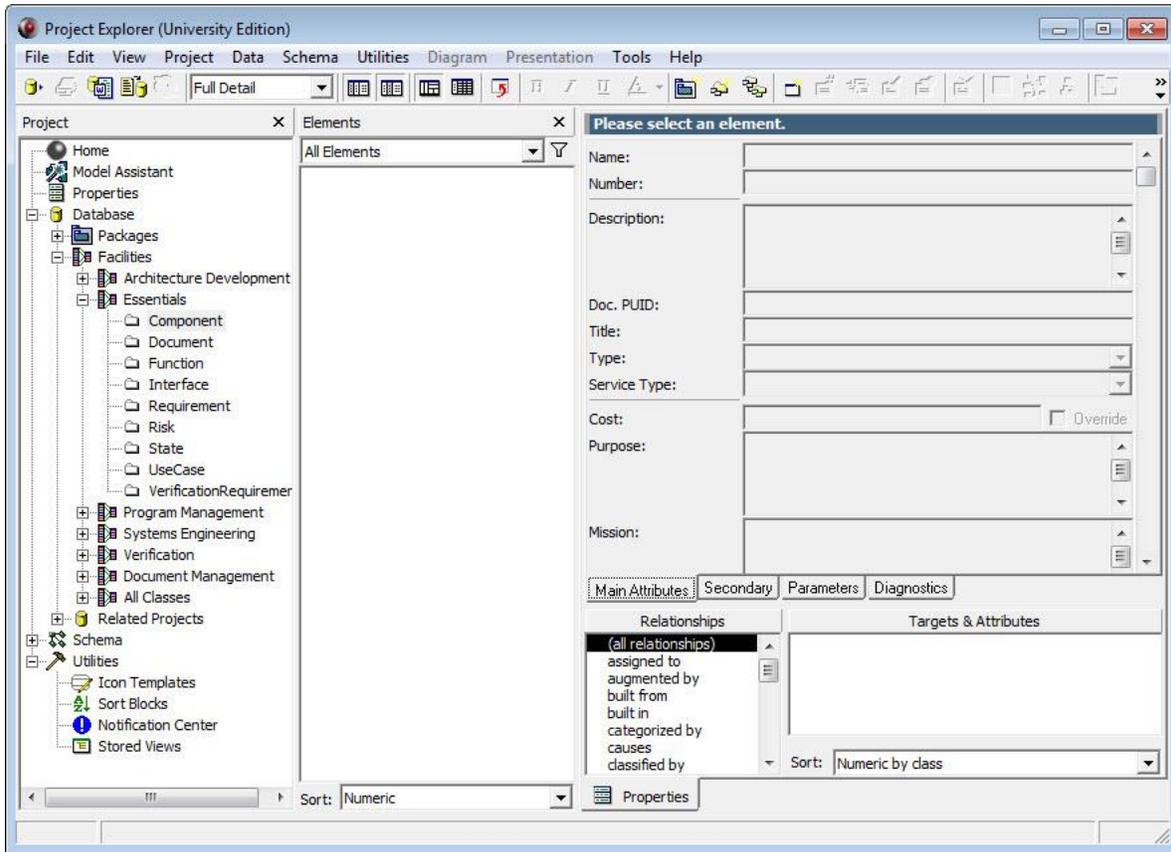
Once you have installed CORE University Edition and obtained an activation key, launch the CORE University Edition.

- Click the Microsoft Windows *START* button.
- Select the *All Programs*, proceed to the *CORE 9 University* submenu, and click *CORE 9 University*. While CORE is being loaded, you will see the following screen.



- You will next see a reminder indicating the time remaining under the current licensing agreement.

When you have successfully opened CORE, the Project Explorer window will open to the default empty project with the Project pane displayed on the left and a Welcome display on the right.



One and only one project is available in CORE University Edition. Clicking on a class folder in the Project pane enables you to start to enter appropriate information into CORE. However, for the purposes of the guided tour, we will import the Geospatial Library sample project into CORE.

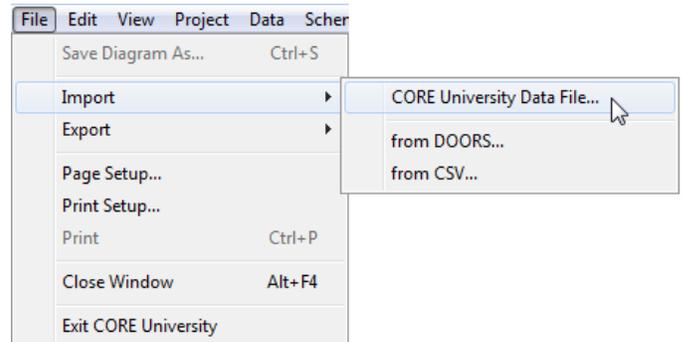
## Importing a Data File

The Geospatial Library sample project is found in the CORE Samples directory. In general, the import/ export capability of CORE allows you to transfer a CORE project from one computer to another or to make a backup copy of the data. Here, we want to import a project so we can see the CORE system design repository populated with data. The project name will not be changed from Default by the import operation.

To import a CORE Data File:

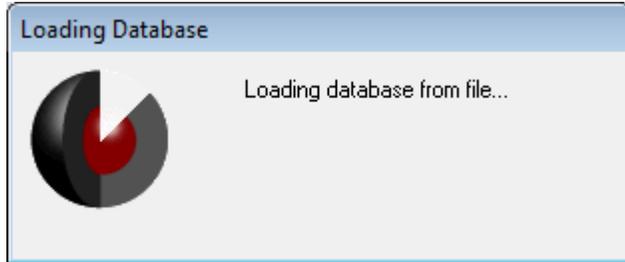
- From CORE's Project Explorer drop-down menus, select **File > Import > CORE University Data File**.

This opens the Import File dialog.



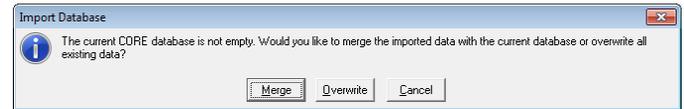
- Navigate to the *CORE 9 University Edition\Data\Samples* directory and select the file named **GeospatialLibrarySampleSolution.a90**.
- Click *Open*.

The file loading progress window is displayed.

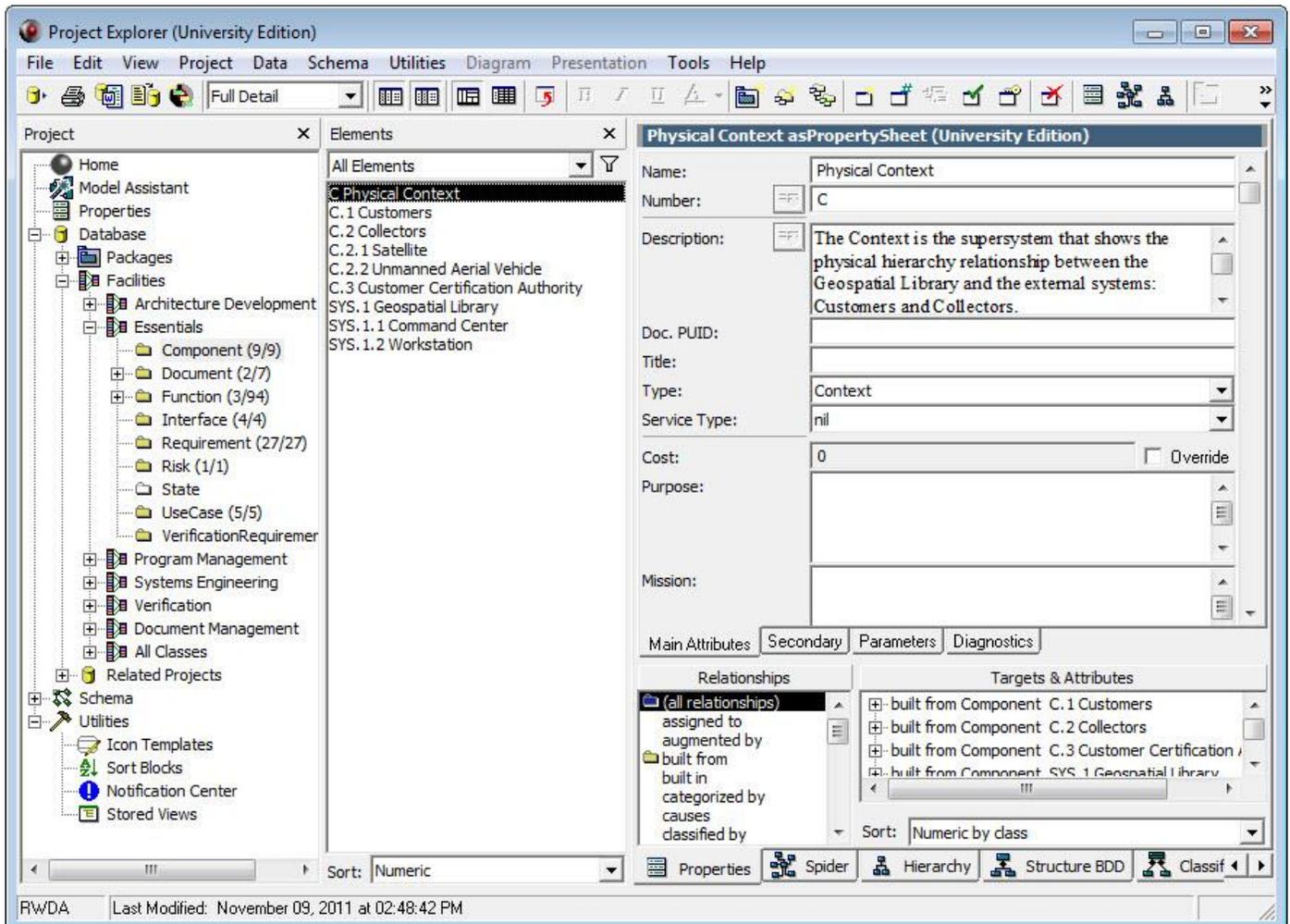


After the database import finishes, the CORE design repository contains the Geospatial Library model.

If you import into a project that has data in it, then the following message appears, allowing you to merge the data save over the data or exit.



- Answering *Merge* will merge the imported data with existing data.
- Answering *Overwrite* will erase the system design repository before beginning the import of the new data.
- Answering *Cancel* will stop the import and return you to the previous view of the CORE database.

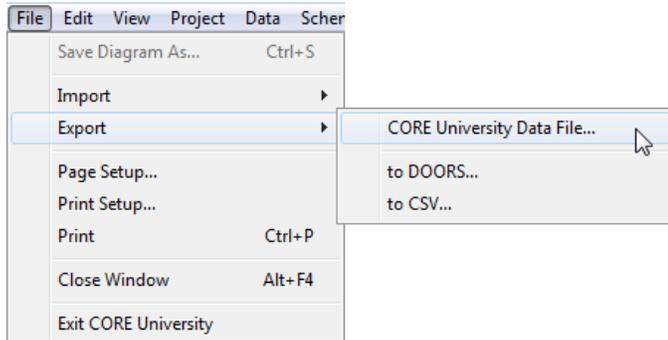




## Exporting (Saving) a Data File

Exporting data is the only permitted option for saving project data in CORE University. To save your CORE data (exports with a .a90 file extension for the University Edition):

- Select *File > Export > CORE University Data File* from the Project Explorer drop-down menus.



The Save File dialog will open.

The suggested name for the project being exported is **Default** and the suggested folder is the CORE 9 University\Data folder. The filename can be changed in this Save File dialog. The .a90 file extension is indicative of a special binary file format used in CORE University Edition.

- Enter your preferred project name and click *Save*.

To clear all existing data and start fresh with an empty database:

- From the Project Explorer menu, select *Project > Erase > Database*.
- Answer *Yes* to the warning message.

## CORE 9 University Edition Features

The CORE 9 University Edition is identical to the commercial version of CORE with the following exceptions:

- The license will expire in alignment with the instructor's class schedule.
- Users are limited to a single project. Therefore, the *Open Project* and *New Project* commands are disabled in the University Edition. When starting in the CORE 9 University Edition, users are already in an empty project and can start adding data to the engineering repository by double-clicking on one of the classes in the left-hand Project pane.
- The capability to save a repository file, which eliminates the import/export cycle, has been disabled.
- The University Edition will import/ export CORE database files in binary format (.a90 file extension) instead of XML format.
- The capability to maintain a recovery log has been disabled.
- The available schema is determined according to the course you are taking. Systems Engineering students automatically use the base Systems Engineering schema. Architecture students use the DoDAF schema.
- DOORS Connector is disabled.
- CSV export is disabled.
- In standard University Edition, there is a total element limit of 200 per class.
- The User/Group Tool, which allows multiple users and groups to be created within the CORE environment, has been disabled.

THIS PAGE INTENTIONALLY BLANK



## **Vitech Corporation**

2270 Kraft Drive, Suite 1600  
Blacksburg, Virginia 24060  
540.951.3322 FAX: 540.951.8222  
Customer Support: [support@vitechcorp.com](mailto:support@vitechcorp.com)  
[www.vitechcorp.com](http://www.vitechcorp.com)