Systems Engineering
Guided Tour
1 Getting Started with CORE ........................................... 1
The Benefits of CORE .................................................. 1
Installing CORE .......................................................... 2
Overview of the CORE Product Family ............................. 2
Key Concepts ............................................................... 3

2 Examining CORE ...................................................... 5
The Sample Problem: The AutoLink System ...................... 5
Guided Tour Conventions ............................................. 5
Getting Started with CORE — Opening CORE ................. 6
Importing CORE Data .................................................. 7
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 ........................................... 14

3 Starting a CORE Project—Extracting Requirements ............ 16
The Sample Problem: The AutoLink System ...................... 16
Summary of the Typical Top-down Process ....................... 16
Capturing the Problem and the Source Requirements .......... 16
Capturing the Document Element ................................. 17
Saving the Document Element in the Design Repository .... 18
Extracting Originating Requirements ............................... 19
Extracting the Top-Level Requirement ............................. 20
Defining a Relationship ............................................... 21
Extracting the Child-Level Originating Requirements .......... 23
Viewing the Requirements in the Element Table ............... 24
Viewing a Requirements Diagram .................................. 25
Viewing a Traceability Hierarchy Diagram ....................... 26
Adding Elements in a Traceability Hierarchy Diagram ......... 27
Viewing the Expanded Traceability Hierarchy ................... 27
Completing the Requirement Definitions .......................... 29
Capturing Requirements Issues ................................... 30

4 Defining the System and its Boundary ............................ 32
Defining the System .................................................... 32
Creating External Systems .......................................... 33
Defining the Context Component ................................... 35
Adding Detail to the External Systems ............................. 36
Viewing a Block Definition Diagram ............................... 37
Creating Function Elements ......................................... 38
Establishing Root Functions for the System/Context/External Systems .................. 39

5 Building the Behavior Model ....................................... 40
Building a Functional Model ........................................ 40
Inserting a Parallel Structure ........................................ 40
Adding Functions to an FFBD ........................................ 41
Adding Inputs and Outputs .......................................... 43
Completing the N2 Diagram .......................................... 45
Viewing an IDEF0 Diagram .......................................... 46
Deriving the Functional (Behavior) Model for Our System .... 47
Adding Inputs and Outputs in an EFFBD or Activity Diagram ...... 50
Completing the Behavior Model ..................................... 52
Completing Traceability ............................................... 54

6 Completing the Physical Model ................................. 56
Extending the Component (Physical) Hierarchy ................. 56
Allocating the Functions ............................................. 57
Completing the Physical Model ..................................... 58
Viewing a Block Diagram ............................................ 60
Impact Analysis ......................................................... 61
Ensuring Full Traceability from Source Document to Physical Architecture .............. 62

7 Generating Documentation ......................................... 63
Generating a System Description Document (SDD) ............. 63
Generating an HTML Report ........................................ 65
Standard Reports Provided with CORE ........................ 68

Appendix—Using CORE University Edition ...................... 70
Opening CORE 7 University Edition .................... 70
Importing a Data File ................................................ 71
Exporting (Saving) a Data File ..................................... 73
CORE 7 University Edition Features ......................... 73
Standard Reports Provided with CORE 7 University Edition .... 74
This page intentionally left blank.
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 7 Guided Tour. This guided tour is intended to familiarize you with basic features of CORE 7, 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 issues. 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 initial 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 7 as well as CORE 7 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 AutoLink System, as it has been captured in the tool. The sample solution is presented in the data file AutoLinkGuidedTour.xml in the Samples directory. You can use this guided tour to recreate the solution yourself from scratch as we go along (starting on page 16).

- Online help and the CORE System Definition Guide that are installed with CORE 7 (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 overview or a hands-on class in CORE, Vitech offers a course to meet your need. Please contact Vitech or visit our website (http://www.vitechcorp.com) for more information on our training classes.

---

1 The file extension for the University Edition data files is .a70
Installing CORE

If you haven’t yet installed the CORE software, please download the appropriate installation guide from our Information Center located on the web at 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 7 product(s), contact us:

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

Overview of the CORE Product Family

CORE Essentials
CORE Essentials provides individuals, small teams, and distributed users a complete model-based system engineering (MBSE) development solution ready for immediate use against a locally installed and maintained repository – yet also adds the ability to connect to a collaborative design environment as required. 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 delivers comprehensive support for DoDAF 2.0 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, Spectrum provides the ultimate answer in capability and flexibility.

CORE Server
Part of a large, complex, or data-rich effort? Add a CORE Server as a remote repository and enable CORE Essentials and CORE Spectrum users the ability to operate offline and independently or as part of the collaborative engineering team operating together in one concurrent database. 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.

The CORE Server provides transaction management between Essentials and/or Spectrum users and the database management system. It includes the administrative functions for managing user accounts, assigning user permissions, and managing the licenses that govern tool utilization. CORE Server includes a dynamic updating mechanism that synchronizes active clients, allowing changes published by one user to be automatically reflected in all clients looking at the same data.

As the engineering team centerpiece, CORE Server offers unparalleled ease of use, team-wide system-level insight, comprehensive analysis, and instant, thorough documentation.

CORE2net
CORE2net turns your CORE Server environment into a web server. As a separately licensed component of the CORE Server, CORE2net 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 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 repository is structured by the SDL which is user extensible, if needed. SDL is an Element-Relationship-Attribute (ERA) language augmented by graphical structures with semantic meaning. SDL is based on the following primitive language concepts:

- **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.

- **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 (the allocated to relationship), CORE automatically creates the performs relationship linking the elements in the reverse direction.

- **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.

- **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.

- **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 COREsim. COREsim dynamically interprets a behavior model so the simulation is always synchronized with the current model contained in the system design repository.

The 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. The diagram below illustrates a subset of the basic schema, showing some of the primary systems engineering classes and relationships between them.
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 repository and, conversely, a change made to the database is automatically reflected in the views.

CORE delivers a mixture of structured and object (SysML) representations enabling you to satisfy the specific needs of your project. CORE provides the following diagrams of interest to engineering and management personnel permitting system models to be viewed in as many layers of abstraction as necessary to understand the model:

- Element Relationship (ER) Diagrams: display the element and its relationships to other elements
- Hierarchy Diagram: 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
- Enhanced Functional Flow Block Diagrams (EFFBD) and Activity Diagrams: portray behavioral flow, control logic, and inputs/outputs/triggers
- Integration Definition for Function Modeling (IDEF0) Diagrams: show functions, inputs, outputs, controls, and mechanisms
- N2 (N-squared) Diagrams: display functions and their internal and external inputs/outputs 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 issues) 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 fifty standard utilities, queries, and report templates 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 repository to gather data and how to 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.

2 A list of the full set of reports provided with CORE is shown on page 68. A limited set of reports is provided with CORE 7 University Edition. Refer to the Appendix for details.
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: The AutoLink System

In this section, we will use a sample database for the AutoLink System 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.

The AutoLink System provides emergency support for the occupants (driver and passengers) of an AutoLink-equipped motor vehicle. The AutoLink System uses Global Positioning System (GPS) to provide current navigation and vehicle location information to the occupants. In the event of an accident, AutoLink automatically contacts AutoLink’s customer support via cellular communications to request roadside emergency help. Customer support transmits vehicle location information, received from the AutoLink vehicle unit, to appropriate emergency response support.

Guided Tour Conventions

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

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>Classes</td>
</tr>
<tr>
<td>Number</td>
<td>Attributes</td>
</tr>
<tr>
<td>allocated to</td>
<td>Relationships</td>
</tr>
<tr>
<td>Contact</td>
<td>User selections and Inputs</td>
</tr>
<tr>
<td>Emergency</td>
<td></td>
</tr>
<tr>
<td>Response</td>
<td></td>
</tr>
<tr>
<td>File &gt; Import</td>
<td>CORE commands, buttons, icons or tabs</td>
</tr>
</tbody>
</table>

Driver communicates with AutoLink customer support via cellular communications for roadside emergency help.
Getting Started with CORE – Opening 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 70 to learn how to get started with CORE.

Once you have installed CORE, launch the CORE 7 application.

- Click the Microsoft Windows START button.
- Select All Programs, proceed to the CORE 7.0 menu, and click Empty Repository. While CORE is being loaded, you will see the following screen.

When you have successfully logged into CORE, the Project Explorer window will open with the welcome display. Once we have loaded data within CORE, we will examine the Project Explorer window in-depth.

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). Therefore, when you open a CORE repository file, you must select a project to work in. You can select an existing project or create a new project either directly by selecting New Project and entering a project name or indirectly by importing an existing project.

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 AutoLink System sample project that was created in CORE and then exported to the CORE 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.

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 70 to learn how to import CORE data.

To import a CORE Data File:

- From CORE’s Project Explorer drop-down menus, select File > Import.

This opens the Import File dialog.

**NOTE**

By default, the Import File dialog directs you to the Data directory since this is typically where data files are stored.

- Navigate to the CORE 7\Data\Samples directory and select the file named AutoLinkGuidedTour.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**.

We will now open the newly created project.

- Select **Open Project** in the Project Explorer and double-click **SAMPLE: AutoLink System (Base Systems Engineering Schema)** 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.
Element Browser

In the Element Browser you can view the structure of the data, create new elements, and update elements. 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. 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 subfolders.

- Select **Function** in the Project pane.
- Next select the **Function** named **Alert Customer Support** in the Elements pane. The Property Sheet for the **Function** will be displayed. Notice that the toolbar now displays an icon for each view that can be displayed for a **Function** and that the view tabs display the same icons.
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 class Named Function, so only those attributes and relationships that pertain to Functions are displayed.

- Open the Property Sheet as a separate window by double-clicking on Alert Customer Support 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 allocated to. 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.
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 a view tab 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. Clicking a view icon on the toolbar opens the view in a separate window.

Let’s look at an example. An Entity Relationship (ER) Diagram graphically depicts the relationships that have been established between the subject element and other elements in the system design repository.

- Select Alert Customer Support from the list of elements in the Function class.
- Click the ER tab to display the ER Diagram of the element (or click the ER icon to open an ER Diagram in a separate window).

All diagram views have a Diagram Elements pane that displays beneath the diagram and can be shown or hidden using the Diagram Elements Table icon. The Diagram Elements pane 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.

- Close the ER View if you opened 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.
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.

- 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 automatically 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.

**NOTE**
Double-clicking on an element in any diagram view opens that element’s Property Sheet in a separate window.
Saving CORE Data

Vitech 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 73 to learn how to export/save CORE data.

To save your CORE data to an XML file:

- Select File > Export 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.

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 73 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.c70 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 automatically 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 the 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.
Starting a CORE Project—Extracting Requirements

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: The AutoLink System

In the rest of this guided tour, you will use CORE and several of its features by systematically building the sample design for the AutoLink System. Please review the AutoLink System 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 AutoLink System 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 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 (functional) model while extending the physical architecture and allocating all behavior onto the physical architecture.

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

Capturing the Problem and the Source Requirements

Now that you have seen how to import a database file and view data, 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 14).
We will start by capturing the relevant information from the source document in CORE. CORE’s Element Extractor is an easy way to transfer text from a document into the CORE database structure.

- In the Project Explorer, click the **Element Extractor** icon or select the **Tools > Element Extractor** command.

Once the Element Extractor opens, you need to load your source document. The source document must be either a DOC, RTF, HTML, or TXT file.

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

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

- Navigate to the **Samples** subdirectory.

**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.

- Select **AutoLinkSourceDocument.doc**.
- Click **Open**.

### Capturing the Document Element

The AutoLinkSourceDocument.doc file displays in the left-hand pane of the Element Extractor while the element being extracted/created is built and defined in the right-hand Element Definition pane. The attributes and relationship fields vary depending on which class is selected from the **Class** pull-down selection list contained in the Element Definition pane.

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

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

Many of the attributes for an element can come directly from the source document. In general, to move text from the left pane to a specific element attribute field on the right, highlight the desired text in the left pane and press the transfer button corresponding to the desired attribute.
To add a **Description** to the **Document** element:
- In the left pane, highlight the first paragraph under **SCOPE**.
- Click the **Description** transfer button in the right pane to transfer the selected text to the **Description** attribute field.
- In the **Name** field, type **AutoLink System**, or highlight the text **AutoLink System** in the left pane and click the **Name** transfer button in the right pane to transfer the text to that attribute field. Add the words **Requirements Document** to the **Name** field.

The **Document Type** attribute is set by choosing from a predefined list of possible values (an enumerated list).
- Click on the **down arrow** next to the **Type** field, and choose **System Requirements Document** to reflect the source of this information.

---

### Saving the Document Element in the Design Repository

Once all desired attributes are defined, we need to enter the **Document** element into the system design repository.
- From the Element Extractor window, click the **Create Element** icon on the toolbar to enter the element 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.

We have finished defining the attributes of the **Document** element named **AutoLink System Requirements Document**. In the next few pages, we will define relationships between several elements as we extract the source requirement elements.
Extracting Originating Requirements

Our next step is to extract and define our **Requirement** elements from the source document, AutoLinkSourceDocument.doc. Notice that after we saved the **Document** element to the system design repository (i.e., clicked the **Create Element** icon), the attribute data is still displayed in the fields. This saves you from re-entering identical attribute values. For the element we are creating here, we won’t reuse any attribute values so we will want to clear the attribute fields before proceeding.

- Click the **Reset Attributes** icon on the toolbar (or select **Extractor > Reset Attributes** from the menus) to clear previous data from the fields.
- From the **Class** selection list, select **Requirement**.

The attribute and relationship fields are now updated to reflect the attributes and relationships for a **Requirement** element.
Extracting the Top-Level Requirement

During this extraction process, we want to establish a hierarchy of requirements. In our case, the Specific Requirements element is refined by a number of subordinate requirements. We will first establish a parent, or top-level, Requirement from the Specific Requirements section of the AutoLinkSourceDocument.doc file. You will need to scroll down in the left pane to see this section of the document.

- Highlight the words Specific Requirements from the text in AutoLinkSourceDocument.doc.
- Click the Name transfer button to insert the text into the Name field, which saves you from having to type the name in the field.
- Type ORD.1 in the Number field. (There is no short way to enter this data.)
- Highlight the text in AutoLinkSourceDocument.doc within the section Specific Requirements, as shown in the figure below.
- Click the Description transfer button to insert the text into the Description field.
- To be thorough, you should enter data in the other attribute fields as appropriate (see Type, Origin, Paragraph Number, and Paragraph Title fields).

ORD.1 represents a user defined hierarchical number. Each cell of a hierarchical number must be either all alphabetic or all numeric characters. Cells are separated by “.” (periods).
Defining a Relationship

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

- Double-click the documented by relationship in the Relationship pane (or select 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).

It lists the allowable target classes for the specified relation, which in this case is documented by. With a Target Class selected, its possible targets are listed. To enter an element into the Targets pane, select the element in the Possible Targets pane and click Add (or double-click on the element). As soon as the target is added, the new target is added to the target list in the right-hand pane. It automatically creates the complementary relation linking the target with the selected elements. You can also click the New button (or double-click on the Class) to create a new target, if you want one that does not already exist.

Here, we will use the source document element we have already created.

- From the Possible Targets pane, select AutoLink System Requirements Document.
- Click the Add button to add the document element to the targets list.
- Click OK to close the Edit Targets dialog.

Returning to the Element Extractor, notice that a target is now associated with the documented by relationship. We are now ready to enter this Requirement in the repository.
• Click the **Create Element** icon on the Element Extractor toolbar. You can also select the **Extractor > Create Element** command or enter **Ctrl+E** to create the new element. The element definition is added to the repository.

Notice the folder icon indicating that a target has been added.
Extracting the Child-Level Originating Requirements

Now that we have defined a top-level Requirement, we will break out each of the six specific requirements to create six individual, child-level Requirements. Each child requirement refines its parent, which we named Specific Requirements. Traceability back to the source document, AutoLink System Requirements Document, is achieved through the parent (as defined with the documented by relationship).

To begin, we took the entire Specific Requirements section of our source document and placed it in a Requirement class element that we numbered ORD.1. This will serve as our parent level. Now we will break up the section, making each numbered requirement its own element of the Requirement class; this will serve as our child level.

We will give each of the six requirements a number and name, specifically:

- ORD.1.1 Impact Detection
- ORD.1.2 Emergency Personnel Contact
- ORD.1.3 Navigation Position Assistance
- ORD.1.4 Wireless Communications
- ORD.1.5 MTBF/MTTR
- ORD.1.6 Availability

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

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

Now we have a clean slate to begin transferring the specific text needed to create the first of our six child elements.

- To extract the first child requirement, complete the following steps:
  1. Type a name in the Name field. Using the list above, the name of the first child element is Impact Detection.
  2. Establish a hierarchical numbering system for the child elements. Number the first element ORD.1.1. (The second will be ORD.1.2 as shown in the list to the left.)
  3. Highlight the text for the description from the source document. For the first child element, we highlight the text in the first paragraph of Specific Requirements.
  4. Click the Description transfer button to place the text in the Description field.
  5. Set the Origin field to Originating, the Paragraph Number field to 3.1, and the Paragraph Title field to Specific Requirements.

- Establish the refines relationship to its parent, Specific Requirements:

  6. Double-click refines from the Relationships pane to open the Edit Targets dialog.
  7. Select Requirement from the list of Target Classes.
  8. Double-click ORD.1 Specific Requirements from the Possible Targets pane to add it as a target.
  9. Click OK to close the Edit Targets dialog.
  10. In the Element Extractor, click the Create Element icon to store this element definition.
Starting a CORE Project—Extracting Requirements

User Tip
An alternative method to add columns is to right-click anywhere in the table, choose Edit Columns. Then in the dialog box, click the **Edit Columns** button. DO NOT reset attributes NOR clear the relationships. Most of the attributes and the relationship you established for the first child element will be used for the subsequent elements.

- Create the other five child elements by repeating steps 1-4 and the step 10 using the **Numbers** and **Names** from the previous page. You can skip steps 5-9 each time through since the same attributes and relationships apply.

### Viewing the Requirements In the Element Table

Let's check what we have done by looking at these elements in the Element Table view.

- We can close the Element Extractor window; we are done with it for now.
- In the Project Explorer, with **Requirement** selected in the Project pane, click the **Element Table** icon.

The Element Table view is convenient for editing attributes and relationships. We will use this view to complete the setting of the **Type** attribute for our child-level Requirements.

**User Tip**

An alternative method to add columns is to right-click anywhere in the table, choose Edit Columns. Then in the dialog box, click the **Edit Columns** button.

- Click in the upper-right corner of the **Type** cell for **ORD.1.1 Impact Detection**. A drop-down arrow will appear. Click on the arrow to access the drop-down enumeration list. Select the **Performance** entry.
- Continue to set the **Type** for each of the children to the values shown below.
Viewing a Requirements Diagram

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

- In the Project Explorer, select Requirement in the Classes pane.
- In the table, click in the Name field of the AutoLink System Requirements Document element to activate the view icons (or highlight the AutoLink System Requirements Document row by clicking in the row number cell).
- Click the Requirements Diagram icon from the toolbar to open a Requirements Diagram.

Your diagram should look similar to the one shown below.
Viewing a Traceability Hierarchy Diagram

In addition to the SysML Requirements Diagram, let’s view a Traceability Hierarchy from our source document. Currently, this will show us traceability from our source document down through requirements. As our model progresses, the diagram will expand to show traceability through the entire design.

Recall that our ORD.1 Specific Requirements is documented by our source Document named AutoLink System Requirements Document. This means that our Document documents our Originating Requirement.

- In the Project Explorer, select Document in the Classes pane.

- In the table, click in the Name field of the AutoLink System Requirements Document element to activate the view icons (or highlight the AutoLink System Requirements Document row by clicking in the row number cell).

- Click 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 AutoLink System Requirements Document was the source from which we extracted the specific requirements, we have traceability from the source through the second-tier (child-level) requirements.

Notice that ORD.1.5 MTBF/MTTR can be broken down further since it combines two requirements. So let’s add another level of Requirements by adding targets for the refined by relationship of ORD.1.5 MTBF/MTTR.
Adding Elements in a Traceability Hierarchy Diagram

CORE allows us to create elements from a diagram. Using the Traceability Hierarchy Diagram, we will create two additional originating requirements as targets of the refined by relationship for ORD.1.5 MTBF/MTTR to break down the requirement to its lowest possible level.

- From the Traceability Hierarchy Diagram, double-click the ORD.1.5 MTBF/MTTR object to open a Property Sheet of this element.

- In the Property Sheet window, double-click the refined by relationship to open the Edit Targets dialog.

- In the Edit Targets dialog, select Requirement from the list of Target Classes.

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

- Double-click Requirement or click New.

- Type a name for the target. We will name one MTBF. Click OK and the new target is listed in the Targets pane.

- From the Edit Targets dialog, double-click Requirement or click the New button again to create another target.

- We will name this one MTTR. Click OK and the new target is listed in the Targets pane.

- Click OK to close the Edit Targets dialog.

- Close the Property Sheet for ORD.1.5 MTBF/MTTR.

Viewing the Expanded Traceability Hierarchy

By default, hierarchy diagrams display 3 levels. A black dot in the upper-left hand corner of an icon, such as ORD.1.5 MTBF/MTTR, indicates that another level can be displayed. In order to see these added elements in our Traceability Hierarchy Diagram, we need to change the diagram to display 4 levels.

- From the Hierarchy Diagram window, click the Diagram Options icon on the upper right-hand toolbar. This will open a Hierarchy Diagram Options dialog.

- Type 4 in the Levels field and click OK.
NOTE
A single icon with a black dot in the upper-left corner can be expanded individually. Select the icon, and from the mouse right-button pop-up menu, select the **Expand Nodes** command and set the number of levels to an integer, the default is one. An icon can also be expanded/collapsed by using the plus/minus icons on the Hierarchy Diagram toolbar or holding down the **Ctrl** key and double-clicking the icon.

The diagram is now updated to display 4 levels. Notice the two new elements are not numbered.

- Click on the **ORD.1.5 MTBF/MTTR** element. Right-click and select **Renumber Element**.

- **ORD.1.5** appears by default in the Renumber Element and Descendents dialog. You can specify how you want to sort elements when renumbering. **Numeric by Class** appears by default.

- Click **OK** to renumber the elements.

Your diagram should look similar to the one shown below.
Completing the Requirement Definitions

The only information that we have entered for each of our new third-level originating Requirements is the **Number** and **Name**. Let us complete their definition by filling in additional attributes.

- Close the Traceability Hierarchy Diagram.
- Click on **Requirement** in the Project pane.
- In the Element Table, fill in the **Description** attributes of our two new **Requirement** elements. You can type in the **Descriptions** or copy and paste the parent’s **Description** into the children and edit the copy.
- Set **Origin** to Originating and **Type** to Performance for both elements.

![Project Explorer (SAMPLE: AutoLink System (Base Systems Engineering Schema))]({})

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ORD. 1</td>
<td>Specific Requirements</td>
</tr>
<tr>
<td>2</td>
<td>ORD. 1.1</td>
<td>Impact Detection</td>
</tr>
<tr>
<td>3</td>
<td>ORD. 1.2</td>
<td>Emergency Personnel Contact</td>
</tr>
<tr>
<td>4</td>
<td>ORD. 1.3</td>
<td>Navigation Position Assistance</td>
</tr>
<tr>
<td>5</td>
<td>ORD. 1.4</td>
<td>Wireless Communications</td>
</tr>
<tr>
<td>6</td>
<td>ORD. 1.5</td>
<td>MTTR/MMTR</td>
</tr>
<tr>
<td>7</td>
<td>ORD. 1.5.1</td>
<td>MTBF</td>
</tr>
<tr>
<td>8</td>
<td>ORD. 1.5.2</td>
<td>MTRR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Composite</td>
<td>Originating</td>
</tr>
<tr>
<td>Performance</td>
<td>Originating</td>
</tr>
<tr>
<td>Functional</td>
<td>Originating</td>
</tr>
<tr>
<td>Constraint</td>
<td>Originating</td>
</tr>
<tr>
<td>Composite</td>
<td>Originating</td>
</tr>
<tr>
<td>Performance</td>
<td>Originating</td>
</tr>
<tr>
<td>Performance</td>
<td>Originating</td>
</tr>
<tr>
<td>Performance</td>
<td>Originating</td>
</tr>
</tbody>
</table>
Capturing Requirements Issues

Now that we have extracted the source Requirements, the requirements 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 Issues. An Issue identifies a problem (as well as its resolution) with an element in the system design or specification. The primary application is documenting problems with requirements.

In our source requirements, we have an explicit availability requirement statement. However, this is not consistent with the availability implied by the MTBF and MTTR requirements, since the availability formula $A = \frac{MTBF}{MTBF + MTTR}$ results in a different availability than specified. Let's add an Issue to ask for clarification on our originating requirements.

- Close all open windows except for the Project Explorer. Set the Project Explorer to the Element Browser view by either clicking on the Element Browser icon or by selecting View > Element Browser.

- In the Project pane, double-click Issue to create an Issue element.

- Name the element System Availability and press OK to close the dialog.

We link an Issue to the element that generated the problem via the generated by relationship. In this case we will link System Availability to three elements.

- Click generated by in the Relationships pane. Add these Requirements to the Targets list:
  - ORD.1.5.1 MTBF
  - ORD.1.5.2 MTTR
  - ORD.1.6 Availability

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

- We will complete the System Availability element to capture the analysis of the problem. Type text in the Description and Alternatives fields. You can use the text as shown below, or develop your own.

Scroll through the list of attributes to see the other fields applicable to an Issue. As a systems engineer, it is important to be as thorough as possible. CORE provides the fields you will need to maintain complete issue documentation.

- 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.
4: Defining the System and its Boundary

Defining the System

In this section, you define your system.

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

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. We will create the Component element of Type System with the Element Extractor because we can use text from the source document and save some typing.

Open the Extractor Window:

- From the CORE Project Explorer, click the Element Extractor icon.
- Click on the Load Document button or select File > Load Document.
- Navigate to Samples\AutoLinkSourceDocument.doc and double-click to load (or click Open).

Create the Element and Define its Attributes:

- From the Class field drop-down list, select the Component class as the destination class.
- Enter AutoLink System in the Name field and SYS.1 in the Number field.
- Extract the second paragraph under SCOPE into the Description field. Extract the third paragraph under SCOPE into the Mission field.
- Set the Type field to System.

Next, we define a relationship by establishing that the Component element is documented by the Document element, which we had named AutoLink System Requirements Document.

Establish a System Relationship:

- Double-click the documented by relationship to open the Edit Targets dialog.
- Double-click AutoLink System Requirements Document from the list of Possible Targets to add it to the Targets pane.
- Click OK to close the Edit Targets dialog.

Save the Element in the Repository:

- From the Element Extractor window, click the Create Element icon to store this Component element definition.
- Close the Element Extractor window.
Creating External Systems

External systems interact with the system element. External systems are represented in CORE’s integrated system design repository by the Component class. We will create four physical Component class elements: Client, Emergency Response, Global Positioning System, and Vehicle. We will also create a logical Component element called AutoLink Context to provide a context to capture how our system interacts with other systems to achieve its objective.

We will use the Project Explorer to create these elements since this data does not come from the source document, i.e., there is no data available to import using the Element Extractor.
Create five new Component Elements:

- In the CORE Project Explorer, double-click Component from the list of Classes to create a new Component element. The New Component dialog prompts for the element name. Pressing the Insert key also allows the creation of a new element.

- Name the element Client and press the Enter key (or click OK.)

- Double-click Component again and create an element named Emergency Response.

- Double-click Component again and create an element named Global Positioning System.

- Double-click Component again and create an element named Vehicle.

- Double-click Component again and create an element named AutoLink Context.
Defining the Context Component

Now we need to define the attributes and relationships of these elements. We capture the context as a Component with Type set to Context. Hierarchies of Components can be constructed using the built from relationship. We want to show that the AutoLink Context is built from the AutoLink System and the four external systems: Client, Emergency Response, Global Positioning System and Vehicle.

Define Attributes and Relationships:

- Open a Property Sheet by double-clicking AutoLink Context.
- Enter the attribute information (Number, Description, and Type) as shown below.
- Establish the built from targets as shown below.
- Close the Property Sheet window.
Adding Detail to the External Systems

To be thorough, you will want to complete the attribute fields for the remaining elements. We will use the Element Table view to complete the element definitions.

- In the Project Explorer, click on the Element Table icon to switch to the Element Table view.
- For the Client, Emergency Response, Global Positioning System, and Vehicle Components, enter the values shown below for their respective Number and Description, and set the Type to External System from the pull-down list for each of the elements, except for Client. Its Type attribute is set to Human.
- When you have filled in the attributes, switch back to the Element Browser view by clicking on the Element Browser icon.
Viewing a Block Definition Diagram

To see the structure (i.e. decomposition) of the AutoLink Context Component element, we can view either a Physical Hierarchy or a Block Definition Diagram. Since earlier we used the hierarchy diagram to show traceability, this time we will select the Block Definition Diagram.

CORE supports both structural (decomposition) and classification (generalization / specialization) Block Definition Diagrams. To see the assembly, we use the structure variant of the Block Definition Diagram.

- In the Element Browser, select the Component AutoLink Context element.
- Click the Structure Block Definition Diagram icon.

By default, the Block Definition Diagram is configured to show the operations and values in addition to the element name. As with all diagrams in CORE, you can adjust the node representation to display as much or as little information as desired by changing the diagram options and the icon representation. In the case of Block Definition Diagrams, teams often display some combination of operations, receptions, parts, ports, and values depending upon the maturity of the model and the specific interest.

- Close the Block Definition Diagram window.
Creating Function Elements

We have defined our Document, our Requirements, our system Component, and the context and external Components. Now we need to add some Functions to describe what our system is to do. A Function is a transformation that accepts one or more inputs and transforms them into outputs. Let us create the root (top-level) function for our system, our context, and each of the external systems. With any system, its root-level function represents the total functionality of the system and its decomposition is represented by its behavioral structure.

To Create Function Elements:

- From the CORE Project Explorer, double-click the Function class to open the New Function dialog.
- Type the name Perform AutoLink Functions.
- Give the element the number 0.
- Type a description for this function in the Description field as shown.
- Create five additional Functions named and numbered as follows:
  - AutoLink Functional Context, number C,
  - Perform Client Functions, number C.1,
  - Perform Emergency Response Functions, number C.2,
  - Perform GPS Functions, number C.3, and
  - Perform Vehicle Functions, number C.4.
Establishing Root Functions for the System/Context/External Systems

We will now allocate the root functions to the components that perform them. This is accomplished by establishing an allocated to relationship.

**Establish an allocated to Relationship:**

- In the Project Explorer, select the Function element **AutoLink Functional Context**.
- Double-click the relationship allocated to to open an Edit Targets dialog.
- Double-click the possible target **C AutoLink Context** to add it to the Targets pane.
- Click **OK**.

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

You are now finished allocating the context function.

- Allocate the other four external root functions (**Perform Client Functions**, **Perform Emergency Response Functions**, **Perform GPS Functions**, and **Perform Vehicle Functions**) to their respective Component elements in the same way. Also, allocate **Perform AutoLink Functions** to the **AutoLink System Component** element. In each case, identify the **Behavior Type** as **Integrated (Root)**.

**User Tip**

An alternative method to create relationships between elements is via drag-drop. If you drag an element from a list onto another element in a list, CORE will prompt you with the possible ways to relate these elements. You may find that you want to open a second Project Explorer for your current project (by using the **Tools>>Project Explorer** command) in order to do this.
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

Next, we begin the behavior analysis of our system. We will do this graphically by building a Functional Flow Block Diagram (FFBD) of our context function, AutoLink Functional Context. CORE’s FFBDs 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.

Open an FFBD:
- In the Element Browser, select the Function element AutoLink Functional Context.
- Click the FFBD icon to open a Functional Flow Block Diagram of the selected element.
- Select the branch between the two reference blocks. This is where we will insert the functional behavior for the context function, the root function for AutoLink Context.

Reference nodes indicate the source and sink of the control flow (what function completed immediately before this flow begins and what function will be enabled when this flow completes). Since there is no source or sink for the AutoLink Functional Context functional flow, the reference node is labeled “Ref.”.

Inserting a Parallel Structure

Recall, we have five functions that we want to include in our diagram:
- Perform AutoLink Functions.
- Perform Client Functions.
- Perform Emergency Response Functions.
- Perform GPS System Functions, and
- Perform Vehicle Functions.

Each of these functions works in parallel. To incorporate these functions into our diagram, we will create parallel branches.
Creating a Parallel Diagram Structure:

- From the FFBD window, click on the Parallel (AND) icon or select the Diagram > Insert > Parallel menu command to insert a concurrency structure.

CORE prompts for the number of branches. We will create a concurrency with 5 branches since we have 5 functions.

- At the prompt, type 5 and then click OK.

The Insert Element dialog works like the Edit Targets dialog described on page 21; it shows the elements that can be inserted. You can select from existing elements or create new ones.

Adding Functions to an FFBD

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

Adding Functions to a Diagram:

- Highlight the top branch.

- In the FFBD window, click the Insert Element icon to open an Insert Element dialog.

The selected function element is added to our FFBD.

- In the same manner, insert Perform AutoLink Functions on the second branch, Perform Emergency Response Functions on the next branch, Perform GPS System Functions on the following branch, and Perform Vehicle Functions on the last branch.

- To remove the reference nodes from the diagram display, click the Diagram Options icon and toggle off Show Reference Nodes.
NOTE
By default, many behavioral views such as the FFBD, EFFBD, Activity, and N2 diagrams display the function’s allocation (the Component to which the Function is allocated) on the bottom of each node. These default icon templates may be changed by clicking the Diagram Options icon.
Adding Inputs and Outputs

Now let’s add inputs and outputs to these root functions (and thereby to the Components that they are allocated to) to show the interdependencies of the Functions. We will do this by using the N2 Diagram (N-squared diagram). An N2 Diagram Shows the data flow between functions. In a later exercise, we will show how to add data flows in an EFFBD.

Open an N2 Diagram:

- In the FFBD window, click on the diagram background to ensure no icons are selected. This will allow us to open a new diagram on the root element.

- Click the N2 Diagram icon on the Views toolbar to open an N2 Diagram of the function AutoLink Functional Context.

Adding an Item:

- On the N2 Diagram, click on the icon Perform Client Functions. The first element selected is ALWAYS the Function whose output from relationship is being set.

- Now, while holding down the shift-key, click to select the Perform AutoLink Functions icon. The shift-clicked second, third, etc., element selected is ALWAYS the Function whose input to or triggers relationship is being set.

- Click on the Connect via Trigger icon or right-click to open a pop-up menu and choose the Connect via Trigger... command. This opens a Connection Dialog.

The N2 Diagram displays the data dimension of the behavior model, whereas the FFBD displays the control dimension of the behavior model. Used in conjunction with an FFBD, the N2 Diagram helps to capture and analyze the functional behavior of your system.
Since the Item element named Client Information Request does not exist, you must create it.

**Creating an Item Element:**

- Type in **Client Information Request** for the name of the new Item element and click **OK**.

The Client Information Request item is added to the N2 Diagram. Notice the direction of the arrow; the Client Information Request item is output from Perform Client Functions and triggers Perform AutoLink Functions.

In an N2 Diagram, the functions appear on the diagonal and items are shown off the diagonal. For a specific function, all input items are located in the function’s column; all output items are positioned in the function’s row.
Completing the N2 Diagram

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

- Using the table to the right as a guide, complete the model by adding three additional **Items**: Emergency Assistance Request, GPS Signals, and Health and Status; and their flow between the functions.

- When you are done, close the N2 Diagram and FFBD windows.

Below is the completed N2 Diagram showing the Item flows between the five functions.

<table>
<thead>
<tr>
<th>Item</th>
<th>output from</th>
<th>triggers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emergency Assistance Request</td>
<td>Perform AutoLink Functions</td>
<td>Perform Emergency Response Functions</td>
</tr>
<tr>
<td>GPS Signals</td>
<td>Perform GPS System Functions</td>
<td>Perform AutoLink Functions</td>
</tr>
<tr>
<td>Health and Status</td>
<td>Perform Vehicle Functions</td>
<td>Perform AutoLink Functions</td>
</tr>
</tbody>
</table>
Viewing an IDEF0 Diagram

We can also view a Function decomposition using the IDEF0 Diagram. The IDEF0 Diagram provides a representation of functional decomposition that complies with the National Institute of Standards and Technology (NIST), Federal Information Processing Standards Publication 183 (FIPS PUB 183), Integration Definition for Function Modeling (IDEF0).

The IDEF0 Diagram is derived from the functional decomposition inherent in CORE’s behavioral models. It enables users to view the functional decomposition and identifies the Inputs, Controls, Outputs, and Mechanisms (ICOM) associated with each function on the diagram. Thus, CORE provides automated IDEF0 Diagram support for customer reviews and each IDEF0 Diagram is guaranteed to be consistent with the underlying behavior model.

The IDEF0 Diagram is a partial subset of the Enhanced FFBD. The IDEF0 Diagram does not show control constructs and, therefore, does not show sequencing.

- Select AutoLink Functional Context from the Function class.
- Click on the IDEF0 Diagram icon on the toolbar.
- Close the IDEF0 Diagram window and respond No to the prompt informing us that not all IDEF rules regarding number of inputs, etc. have been satisfied.
Deriving the Functional (Behavior) Model for Our System

We will now decompose the root function for our system, **Perform AutoLink Functions**, to add another level of detail. For this exercise, we will use the Enhanced FFBD (EFFBD). Those who favor SysML representations could equivalently use CORE's Activity Diagram. CORE's integrated model-based system allows you to view and edit the model through whatever representation best meets your current need.

While the FFBD allows us to add constructs and functions to the diagram, items cannot be added directly through the diagram. However, the EFFBD (i.e., FFBD with data) allows us to add constructs, functions, and items to the diagram directly and thereby add them into the CORE system design repository.

- From the Project Explorer, select the **Function** element **Perform AutoLink Functions**.

- Click the **EFFBD** icon to open an EFFBD view.

In the same manner that we used with the FFBD earlier:

- Select the branch between the reference nodes at the insertion point.

- Insert a **Parallel (AND)** construct with four branches.

- Annotate each branch by double-clicking on each branch and typing the annotation: **Navigation**, **Itinerary**, **Vehicle Condition**, and **Customer Support**.

- Hide the Reference Nodes to display the diagram as shown (i.e., click the **Diagram Options** icon and toggle off **Show Reference Nodes**).

Next, we will add functions to each branch.

- Select the top branch labeled **Navigation**.

- Click on the **Insert Element** icon (or select **Diagram > Insert > Element**).

- From the Insert Element dialog, create two new **Functions** in the following order: **Compute Latitude/Longitude** and **Integrate Current Position and Map**.

- Click **OK** to add these functions to the diagram serially in the order they were added to the target list.

In a similar manner:

- Insert the new function **Calculate and Display Route** to the **Itinerary** branch.

- Insert a new function named **Assess Vehicle Health** to the **Vehicle Condition** branch.

- On the **Customer Support** branch, insert three new functions in the following order: **Identify Client and Process Request**, **Assess AutoLink Sensor Data**, and **Contact Emergency Response**.
To complete our control logic, we are going to add an **Iterate** construct to the **Navigation** branch. An **Iterate** construct is a shorthand construct for a repeated sequence; it is controlled by a **DomainSet** which defines the number of times or frequency with which a functional sequence repeats.

- On the **Navigation** branch, select the function **Compute Latitude/Longitude**.
- Click on the **Iterate** icon or select the **Diagram > Insert > Iterate** command.
- When CORE opens the Select Domain Set dialog, either double-click on the class **DomainSet** or click **New** and create a **DomainSet** named **Position Recalculation Rate**.

**NOTE**
Selecting an object (function or logic construct) causes the insertion to be to the **left** of the selection. Selecting a branch causes the insertion to be to the **right** of the selection.
• Select **Compute Latitude/Longitude**.

• Shift-click on **Integrate Current Position and Map**. (Ensure that only these two Functions are highlighted and not the entire branch.)

• From the **Edit** drop-down menu select the **Cut** command (or use the shortcut **Ctrl+X**).

• Select the branch portion within the **Iterate** (i.e., between the IT icons). From the **Edit** menu, select the **Paste** command or use **Ctrl+V**. This will insert the two **Functions** inside the **Iterate**.

---

Now, let’s number the functions on our EFFBD.

• Be sure that nothing is selected by clicking in the background. Then right-click and select the **Renumber Element** command. A Renumber Element dialog is provided.

• You may keep the value suggested or change it to a new value. In this example, select 0 (zero) and click **OK**. A warning message will be presented.

• Click **Yes** in response to that message and then CORE will automatically renumber the functions.
Adding Inputs and Outputs in an EFFBD or Activity Diagram

Inputs, outputs, and triggers can be added directly to functions in the EFFBD or Activity Diagram in the same manner as we used earlier with the N2 Diagram. Flows between functions can be defined by selecting the functions in the order of from-to and using the Connect via Trigger or Connect via Data icons or the Diagram > Connect via... commands. Remember to hold down the shift key when selecting from-to connections.

Individual function inputs, outputs, and triggers may be added with the Edit Inputs (or Outputs, or Triggers) icon or the Diagram > Edit Inputs (or Outputs, or Triggers) command. Note that triggers differ from input items in that they are required to be present before that Function can begin execution.

- Open either an EFFBD or Activity Diagram.
- Add items to the behavior model using the table below. Note that Routing Instructions and Emergency Notification Requests do not yet exist as items and will need to be created.
- After working with your chosen diagram (EFFBD or Activity Diagram), open the other diagram to view its notation.

<table>
<thead>
<tr>
<th>Function</th>
<th>Items to Add</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>triggers</strong></td>
</tr>
<tr>
<td>Compute Latitude/Longitude</td>
<td>GPS Signals</td>
</tr>
<tr>
<td>Integrate Current Position and Map</td>
<td></td>
</tr>
<tr>
<td>Calculate and Display Route</td>
<td>Client Information Request</td>
</tr>
<tr>
<td>Assess Vehicle Health</td>
<td>Health and Status</td>
</tr>
<tr>
<td>Identify Client and Process Request</td>
<td>Emergency Notification Requests</td>
</tr>
<tr>
<td>Contact Emergency Response</td>
<td></td>
</tr>
</tbody>
</table>

In CORE, with every action you are building the underlying integrated model, and CORE’s view generators produce the desired representations. This allows you to switch between representations seamlessly upon demand, increasing the available tools to support your analysis and communication needs.

Icons on diagrams can be moved within their positioning constraints. To move an icon, first select the object. The upper left handle is the move handle. To reposition the selected object, click the mouse within the handle and drag the object to the desired position.

In an FFBD, EFFBD, or Activity Diagram, functions and logic constructs can only be moved horizontally. In an EFFBD, items can be moved both horizontally and vertically.

The items in the following diagrams have been repositioned from their default placement.
Completing the Behavior Model

We will decompose the Function Assess Vehicle Health. We will use an EFFBD to decompose this function, but you can use an Activity Diagram if you wish:

- In the EFFBD, hold down the Ctrl key and double-click on the Assess Vehicle Health object. This will open the same diagram type on the selected object.
- Toggle off the Reference Nodes to display the diagram as shown (i.e., click the Diagram Options icon and toggle off Show Reference Nodes).
- Select the branch and click the Insert Element icon to add a function to the branch.
- In the Insert Element dialog, create a new Function named Process Health and Status Information, adding it to the Selections list.
- Click OK to close the dialog. The new element is now displayed on the diagram.

Next we will add a function to the Accident branch.

- Select the Accident exit branch of the multiple exit function and click the Insert Element icon to add a function to the branch.
- In the Insert Element dialog, create a new Function named Alert Customer Support, adding it to the Selections list.
- Click OK to close the dialog.

Now, we will number the functions on the diagram.

- Click in the blue area of the diagram, so that no object is selected. Then right mouse click and select Renumber Element.
- Click OK to keep the same number for Assess Vehicle Health.

The new functions will be automatically numbered.
Let’s now add **Items** to our behavior diagram.

- Select the Function **Process Health and Status Information**. Click on the *Edit Triggers* icon to add an **Item** as a trigger. In the Edit Triggers dialog, select **Health and Status** and click *Add*. Click *OK*.

- Next select **Alert Customer Support**. Click on the *Edit Outputs* icon to add an **Item** as an output. In the Edit Outputs dialog, select **Emergency Notification Requests** and click *Add*. Click *OK*.

- Close all open diagram windows.
Completing Traceability

Now that we have defined our system and its functional (behavior) model, 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 **ORD.1.1 Impact Detection**. Double-click the basis of relationship to open an Edit Targets dialog.

- In the Edit Targets dialog, select target class **Function**. Select possible targets **Process Health and Status Information** and **Alert Customer Support**. Then click the **Add**. Click **OK**.

- Click on the **Hierarchy** tab in the Element Browser and you will see the Traceability Hierarchy Diagram in place of the Property Sheet. You see the traceability of the **Requirement** down to the **Function** level.
- Returning to the Property Sheet, extend the traceability for the following Requirements:
  - ORD.1.2 Emergency Personnel Contact
  - ORD.1.3 Navigation Position Assistance
  - ORD.1.4 Wireless Communications
  - ORD.1.5.1 MTBF
  - ORD.1.5.2 MTTR

- Use the diagrams below to guide you. When tracing to a Function, use the relationship basis of; when tracing to a Component, use the relationship specifies. (The Requirement ORD.1.6 Availability does not trace to anything in our model at this time.)

**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.
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

Let us now assume that our AutoLink System is built from two components: AutoLink Vehicle Segment and Customer Support Center.

- In the Element Browser, create the two new elements, AutoLink Vehicle Segment and Customer Support Center.

- Assign their respective Number, Description, and Type attributes as shown below and to the right.

- For each component, establish the built in SYS.1 AutoLink System relationship. Note that built in is the inverse of built from.

- Now, open a Physical Hierarchy Diagram of the AutoLink Context component. Do this by selecting the element AutoLink Context. Click on the Hierarchy Diagram icon, select Physical from the definitions drop-down list, and click OK.
• Close the Hierarchy Diagram window.

**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.

• In the Element Browser, establish the *performs* relationships to show that the (leaf-level) *Components*, *AutoLink Vehicle Segment* and *Customer Support Center*, perform the (leaf-level) *Functions* shown in the two custom hierarchies that follow (*performs* is the inverse of the *allocated to* relation).
Completing the Physical Model

By allocating our leaf-level Functions to Components, we have established the interfaces between our Components. We will formalize this in our model by establishing physical connections between our Components. Physical connections are represented in CORE using Link elements.

Confirm your allocations by opening the custom hierarchies shown above.

- Select **SYS.1.1 AutoLink Vehicle Segment** in the Elements pane. Click on the **Hierarchy Diagram** icon to open the Hierarchy Definition Dialog.

- Click on the **More** button to expand the dialog, and then click on **<<Custom>>** in the definitions list to create a new custom hierarchy.

- In the Candidates pane, select performs and click **Add**. This will add performs to the Selections pane.

- Click **OK** and a Hierarchy Diagram window will open displaying the custom hierarchy.

- After examining the hierarchy, close the Hierarchy Diagram window.

- Repeat the above sequence for **SYS.1.2 Customer Support Center** to open, review, and close the custom hierarchy.

User Tip

Pressing the first letter of the desired value jumps to the first item beginning with the entered letter. For example, with a candidate selected, pressing **P** causes participates in to be selected. Pressing **P** again moves to the next item beginning with **P**.
6: Completing the Physical Model

- In the right-hand pane, select transfers and move it to just after the connects to column by using the up arrow. Click OK.

The Elements Table will update to display the new column order.

- By double-clicking on Link in the Project pane, create five Links named as follows:
  - AVS-CSC
  - Client-AutoLink
  - ER-AutoLink
  - GPS-AutoLink
  - Vehicle-AutoLink

- Establish the relationships shown below. The connects to targets are the two Components that a Link connects. The transfers relationship identifies the Items that are passed along the Link. To establish a relationship, place the cursor in the appropriate cell, right-mouse click, and select Edit Targets from the pop-up menu. This will open the Edit Targets dialog.

- Return to the Element Browser view by clicking on the Element Browser icon.
Viewing a Block Diagram

We have defined Components and Links and established their inter-relationships. We can now view this graphically by opening a block diagram.

CORE provides several different block diagram representations. Drawing upon classical systems representations, CORE supports the Interface Block Diagram (displaying logical Interfaces between Components) and Physical Block Diagrams (displaying physical Links between Components). Supporting SysML representations, CORE also delivers two variants of the Internal Block Diagram (IBD) – a Standard IBD displaying Interfaces and a Flow IBD displaying Links.

For this analysis, we will use the Flow Internal Block Diagram which displays linkages, ports, and the items being transferred across the Links.

- Select AutoLink Context from the Component class.
- Click the Flow Internal Block Diagram icon.

In the Flow IBD, the Components are represented by icons and the Links are represented by labeled lines where the label is the name of the Link with the Items being transferred shown in curly brackets. As with all CORE diagrams, this diagram can be edited directly and the information residing in the integrated system design repository will be automatically updated.

User Tip
To export a diagram for use in another application, select File > Save Diagram As… from the Project Explorer or Diagram window.

When an IBD is initially opened, the Component icons and Link lines are in a default placement. You can change the layout of the diagram by doing the following:

- To reposition a Component icon, select the icon and use the top-left handle to move it to the desired location.
- To resize a Component icon, select the icon and use the bottom-right handle to size the icon.
- To reposition a Link line, select the line. Notice that at each end of the line there is a handle that can be moved along the outer edge of the Component icon. Move these to the desired location in relationship to the attached Component icon. If the line is bent, there are additional handles to control the location of the bends.
- To reposition a Link label, click on the label. The lower-right handle controls the dimensions of the label box and the upper-left handle controls the location.
- When you have rearranged the diagram to your satisfaction, close the Flow IBD window.

By default, when Links are connected to a Component via the connects to relationship, the default flow is bi-directional ("inout" in SysML terminology). To change this:

- Double-click on desired Link to open a property sheet.
- Select the connects to relationship.
- In the Targets & Attributes pane, double-click on the Direction attribute and select the desired direction from the drop-down list.
Impact Analysis

To show some of the power of developing our system model in an integrated system design repository, let us illustrate with a typical systems engineering example. Suppose that the customer wants to know the impact of exchanging out/replaceing the **AutoLink Vehicle Segment**.

- Select the component **AutoLink Vehicle Segment** in the Elements pane.
- Click the **Hierarchy Diagram** icon.
- Select **Behavior Impact of Physical Change** from the list of stored definitions and click **OK**.

The diagram below shows which functions and which I/O (inputs and outputs) may be impacted. It shows that the replacement component must perform functions numbered 1, 2, 3, 4.1, and 4.2. In addition, the data interfaces between the replacement component and the other elements of the system/context are those items in the bottom row of the hierarchy.
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 power to see that gold-plating and boilerplate are deleted. Traceability also supports internal consistency, which is difficult to attain when applying traditional techniques on real systems. 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 issues. For example, we see that requirement **ORD.1.6 Availability** has not yet been addressed. A Traceability Hierarchy Diagram does not have to be opened to determine that **ORD.1.6 Availability** has not yet been addressed. This can also be done in the Element Browser using a filter.

- Close the Hierarchy Diagram window.
- Select the class **Requirement** 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.

Let’s open a Traceability Hierarchy Diagram of our source document.

- In the Element Browser, select the class **Document** then select **AutoLink System Requirements Document**. Click on the **Hierarchy Diagram** icon, choose the stored definition **Traceability**, and click **OK**.
- On the diagram toolbar, click the **Diagram Options** icon to open the Diagram Options Window. On the **Options** tab, change the **Levels** to 6; on the **Scale** tab, select **Fit to Window**; click **OK**.
Generating Documentation

In this section, you will generate output from CORE

• Output a System Description Document
• Generate an HTML Report
• 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 models and related information. For each element appearing in the SDD, key attributes and relationship 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.

• 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 AutoLink System 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 AutoLink SDD.rtf.

Press Save.

CORE generates the report as an RTF (Rich Text Format) 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.

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

The System Description Document for this example is approximately 57 pages long.
Generating an HTML Report

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.

- To run this report, click the Run Script icon on the toolbar.

- In the Select Script dialog, select Systems Engineering from the Folder pull-down list. Select HTML Report from the Script pull-down list. Click OK. (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 sort the elements, click Yes.

- When prompted to select the diagram types, click Add All.

- Click OK.

- When prompted to select a custom file, click No.

- When prompted to show blank attributes, click No.

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

- Click OK.
The file 0_homepage.htm will automatically 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 Function link. On the right, a list of all elements in the Function class in your project is displayed.
- Click on 4 Assess Vehicle Health to show the elements information.

**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 4 Assess Vehicle Health to return to the element information. From here on, you will be able to view any diagram in the HTML output without a warning.

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 **Enhanced FFBD**. The EFFBD of the function will be displayed in place of the element attributes and relationships. Each function and item icon is a link to that element.

- 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…** 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 `.c70` 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 7

All of the scripts listed below are included with the full version of CORE 7.

**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)

**System Engineering**
- Attribute History Report
- Baseline Elements
- Element Definition
- Generic Table Output
- HTML Report
- IDEF0 Node Index
- Indented Hierarchy Report
- Project Compare Report
- Purge Attribute History
- System Description Document (SDD)

**Consistency Checks**
- Item Consistency Check
- Keyword Search
- No Descriptions Query
- Open Issues Query
- Recursive Elements Check
- Unallocated Leaf-Level Function Query
- Unaddressed Leaf-Level Requirements Query
- Unverified Leaf-Level Requirements Query

**Utility**
- Assign Documentation PUID
- Element from ID
- Structure Traversal

**Database**
- Database Statistics Report
- Schema Definition Report

**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
• Activities to Operational Tasks Traceability Table
• View from External File

Versioning

• Attribute History Report
• Baseline Elements
• Purge Attribute History
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. 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. 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 7 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 University 7.0 submenu, and click CORE University 7.0. 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. 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 AutoLink System sample project into CORE.

**Importing a Data File**

The AutoLink System 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.*
Appendix: Using CORE University Edition

Navigate to the CORE University Edition 7: Data\Samples directory and select the file named AutoLinkGuidedTour.a70.

- Click Open.

The file loading progress window is displayed.

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 importation of the new data.
- Answering Cancel will stop the import and return you to the previous view of the CORE database.

You can continue with the steps on page 9 to go through the Guided Tour.
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 .a70 file extension for University):

- Select File > Export 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 7 University\Data folder. The filename can be changed in this Save File dialog. The .a70 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.

You can continue with the steps on page 16 to go through the guided tour.

CORE 7 University Edition Features

The CORE 7 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 7 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 an repository file, which eliminate the import/export cycle, has been disabled.

- The University Edition will import/export CORE database files in binary format (.a70 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.

- A subset of the available reports has been provided (see following list). In addition, the capability to create and modify report scripts has been disabled.

- The User/Group Tool, which allows multiple users and groups to be created within the CORE environment, has been disabled.
• An artificial limit on the number of elements in each class has been added. Each class is limited to 200 elements.

• An artificial limit on the number of schema extensions has been added. However, the user is free to create as many additional attributes definitions as desired. The user is limited to:
  o 2 additional classes
  o 4 additional relations
  o 1 additional facility

**Standard Reports Provided with CORE 7 University Edition**

Scripts shown below are available in the University Edition of CORE 7. Refer to page 68 for the complete set of scripts provided with the full version of CORE 7.

**System Engineering**

- Generic Table Output
- HTML Report
- IDEF0 Node Index
- System Description Document (SDD)

**Queries and Consistency Checks**

- Item Consistency Check
- Keyword Search
- No Descriptions Query
- Open Issues Query
- Recursive Elements Check
- Unaddressed Leaf-Level Requirements Query
- Unallocated Leaf-Level Function Query
- Unverified Leaf-Level Requirements Query

**Database**

- Database Statistics Report
- Schema Definition Report

The parsers, utility, linked HTML, formal documentation, and DoDAF scripts are not available in the CORE 7 University Edition.