



New Features Guide

Redefined Usability
Diagram Substance with Style
An Expanded Representation Toolkit
Enhanced Model Navigation
Automated Model Assistance
and a myriad of
additional refinements



Copyright 1992 - 2011 Vitech Corporation.
All Rights Reserved

Copyright © 2011 Vitech Corporation. All rights reserved.

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

Restricted Rights Legend

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7013.

Vitech Corporation
2270 Kraft Drive, Suite 1600
Blacksburg, Virginia 24060
540.951.3322 FAX: 540.951.8222
Customer Support: support@vitechcorp.com
www.vitechcorp.com

CORE® is a registered trademark of Vitech Corporation.

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

Publication Date: October 2011

Contents

Overview of CORE 8.....	1
Interacting with Diagrams – The New Framework.....	2
Drag-Drop.....	2
Moving and Resizing Diagram Content.....	4
Auto-sizing Icons.....	5
Enhancing Model Visualization and Communication.....	6
Graphical Images and Geometric Frames.....	6
Inserting Generic Graphics.....	11
Inserting Generic Shapes.....	12
Font Decorations.....	13
Icon Templates.....	13
Set as Default Icon.....	13
Hiding Diagram Content (Eliding in SysML Terminology).....	14
Presentation Menu.....	15
Additional Diagram Framework Changes.....	15
Expanding Our Toolkit of Representations.....	17
Interface and Physical N2 Diagrams.....	17
Spider Diagrams.....	18
Layouts.....	20
Additional Representation Changes.....	23
Organizing, Navigating, and Focusing on Models.....	24
Packages.....	24
Facilities.....	25
Additional Explorer Content.....	25
Perspectives.....	26
Additional Navigation Changes.....	27
Assisting the Analyst.....	28
Accelerating the Initial Requirements Phase.....	30
A Myriad of Additional Refinements.....	32
Scheduling Tasks.....	32
Polishing Reports.....	33
Audit Logs.....	36
Setting Attributes for Multiple Objects.....	37
Updated Online Help.....	37
Additional Changes.....	38
Expanding the Model.....	40
Migrating Projects from pre-v80 Schemas.....	40
Comprehensive List of CORE 8 Schema Changes.....	41

THIS PAGE INTENTIONALLY BLANK

Overview of CORE 8

CORE provides individuals and teams a complete model-based system engineering (MBSE) development solution. CORE's foundation delivers 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 8 represents a landmark step and truly unlocks the power of MBSE. Combining our historical strengths of an integrated, model-driven architecture with the usability demanded of modern software, version 8 redefines CORE. It delivers the user experience of a technical drawing package to complement the power of a model-driven solution, accelerating system development to think-speed. It redefines communications and analysis by delivering rich and robust representations - substance with style, your information your way direct from CORE. And it lowers the barrier of entry to MBSE, aiding the engineer through automated model assistance.

CORE 8 highlights include:

- **Redefined Diagram Usability** – CORE has long provided the systems engineering power you need. CORE 8 delivers the usability you demand. A completely redefined and re-imagined diagram framework allows you to construct your system models with drag-drop simplicity to complement the richness and consistency of a model-driven framework. CORE 8 reduces your learning curve, improves your productivity, and guides you through MBSE as you learn.
- **Enhanced Model Visualization and Communication** – CORE 8 represents a landmark shift, maintaining all of the power and consistency of the past and complementing it with robust and rich representations. Generate your system diagrams with a blend of traditional geometric blocks and stock and custom graphic. Use rich style options to hide peripheral information and focus attention on content you deem critical. Do it all quickly and easily, delivering enhanced analytical value and customer communications from within your MBSE framework.
- **Expanded Toolkit of Representations** – To address the problems at hand and communicate with a diverse set of specialists and stakeholders, systems engineering demands the richest representation set possible. CORE 8 continues to add to its rich and diverse collection of integrated representations with spider diagrams, interface N2 diagrams, physical N2 diagrams, and new layout options. This blend of structured and free-form representations brings further insight, greater context, and enhanced communications to your efforts.
- **Enhanced Model Organization and Navigation** – With the introduction of packages, CORE 8 complements its traditional model organization and navigation approach with a tailorable framework allowing you to better structure and understand your solution. With perspectives, CORE provides customized – and customizable – lenses to better view your model. The net result is a lower barrier to entry – for MBSE and your system solution.
- **Automated Model Assistance** – “Model Assistant” represents an innovative new capability in CORE 8. The automated model assistant reduces the systems engineering learning curve while automating key tasks to better align your logical and physical architectures. The model assistant is a next step in delivering the value-add of a true model-based approach.
- **Accelerated Requirements Phase** – Combining a new automated parser with critical human-in-the-loop capabilities, CORE 8 reduces the perspiration of project startup while allowing you to focus on analysis, comprehension, and understanding. Gain command of the problem quickly and demonstrate rapid time-to-first-value for your team and your customer.
- **Expanded Models** – With CORE 8, we continue to enhance the underlying schemas to reflect our lessons learned and best practices in the application of model-based systems engineering. Updates to our base and DoDAF schemas give additional power in areas of change management, test and evaluation, and program management.
- **Additional Refinements** – CORE 8 brings a long list of enhancements and refinements to the feature set, including the ability to schedule tasks for automated execution, the introduction of audit logs, the ability to set attributes for multiple objects simultaneously, and a completely rewritten help file to serve as a valued resource. Other changes include refinements to project administration abilities, import / export, user preferences, and much more.

Interacting with Diagrams – The New Framework

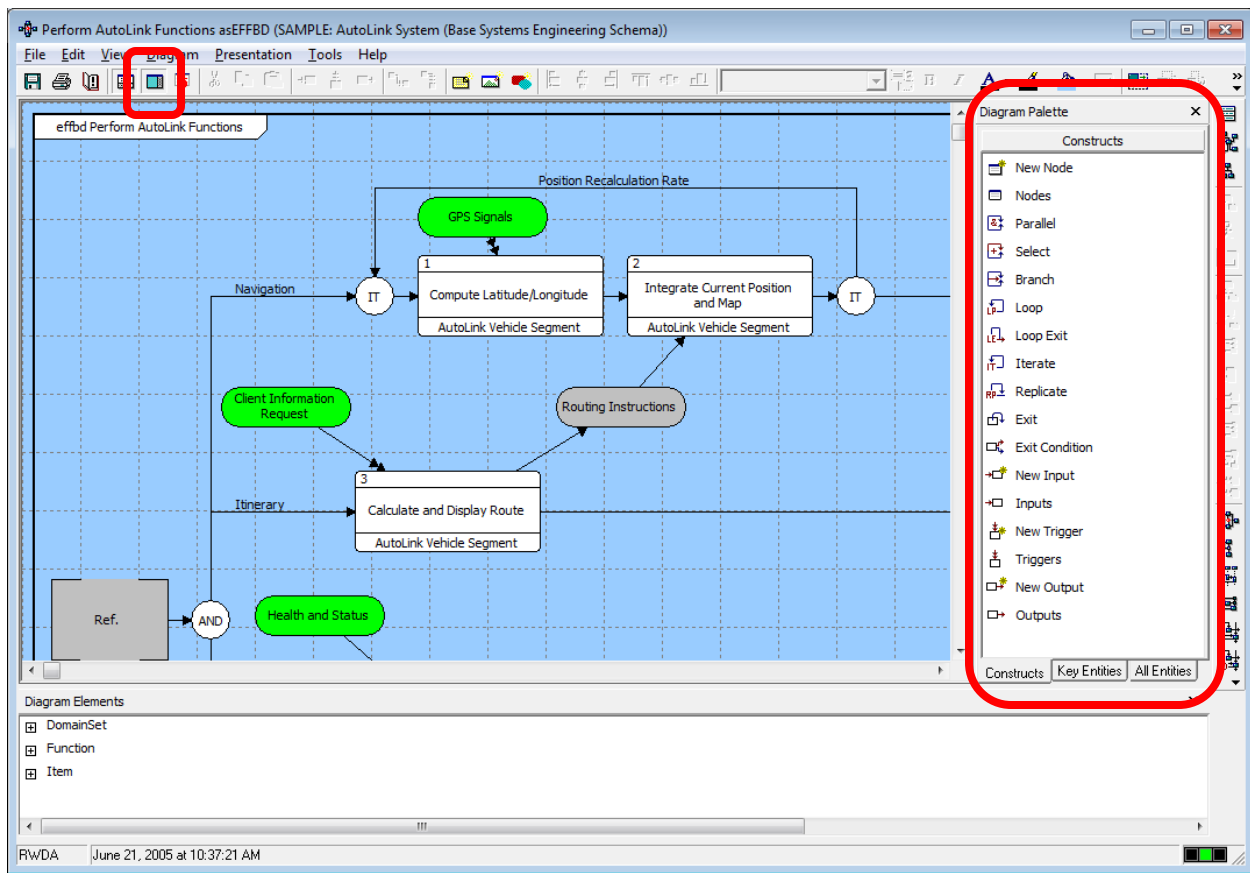
CORE features two primary interfaces – the project explorer for creating and defining elements and the myriad of diagrams for visualizing and developing the system definition. In enhancing the diagram framework, we are seeking to recreate the positive user experience of a technical drawing tool such as Visio® while providing the correctness, consistency, and power of a model-driven representation.

Drag-Drop

In CORE 8, virtually anything you want to do to visually develop your system definition can be done via drag-drop. Drag an element from a list onto a node to create a relationship. Drag two diagram nodes onto one another and have CORE guide you towards the relationships that make the most sense for the specific diagram. Drag a construct from a palette onto an activity branch to insert it. Repackage your system logic by dragging a construct from one branch to another or from one diagram to another. While all of the menus and toolbars remain for comfort, consistency, and a reminder of what is possible, we expect most users to quickly gravitate to manipulating the diagrams via drag-drop.

Diagrams now include a palette along the right edge of the screen. This palette can be toggled on or off to ease model development or maximize screen space using the Hide / Show Diagram Palette toggle in the toolbar (circled in red in the following figure). Each diagram contains up to three tabs on the palette:

- **Constructs.** For all database-side diagrams with defined semantics (everything except hierarchy, spider, and ER diagrams), the constructs pane provides a ready palette of diagram constructs to drag onto the diagram. Dragging a construct onto a branch inserts the construct at the point you release it. Dragging a construct onto a node creates a relationship. As you drag, the cursor changes to show you if the construct can be dropped and the target emphasis highlights where the content will be inserted. For constructs that require additional information (what specific function you wish to insert, what domain set will be used, etc.), when you release the mouse to drop the construct, CORE will prompt you for the remaining information. (And if you want to insert based upon the traditional CORE rules, just double-click a construct on the palette.)
- **Key Entities.** Most diagrams focus on specific aspects of the model – the physical composition and linkages on a physical block diagram; allocated functions, their triggers, and outputs on a sequence diagram; etc. To focus attention on these key classes and simplify navigation, the Key Entities tab shows just these classes. For example, on an internal block diagram, drag a component onto the background to add it to the physical decomposition. Drag an interface onto a diagram node to make the connection.
- **All Entities.** While each diagram focuses on specific model aspects, we are always looking at representations of the real underlying model. In many cases, we may want to create a relationship immediately – even if it won't be shown on this particular view – rather than flipping from view to view. For this reason, we have added the All Entities tab. Drag any element onto a node, and CORE will prompt you with the possible ways to relate the two elements. Not only does this ease manipulation, it significantly lowers the learning curve on the language of MBSE.



In implementing drag-drop, we have adopted the following standards. These reflect a mix of standard Windows approaches (for ease of adoption) with specific CORE conventions (for maximum usability).

- Within a diagram, if you right-click and move the mouse, CORE will initiate a drag-drop operation for the current selection(s). Alternatively, if you hold down the control key and then left-click, CORE will initiate a drag-drop operation. Note that you can release the control key as soon as you left-click. You do not have to keep the control key depressed throughout the entire operation.
- If you right-click to drag, when you release the right mouse button, CORE will prompt you with a list of available operations – move, copy, or link. The actual operations shown will change based upon the combination of objects being dragged and where you choose to drop them. CORE automatically determines the appropriate semantics and presents you with those choices.
- If you left-click to drag, the operation to be performed will change based upon the key combination you have depressed when you drop the object.
 - If no keys are depressed, CORE will implement its default operation for the specific combination of objects.
 - If the control key is depressed when you release the button, the operation is a copy.
 - If the shift key is depressed, the operation is a link.
 - In no circumstance will CORE present you with a choice that is not valid. If the CORE semantic for the drag-drop combination is limited to a link operation, holding the control key down will not allow you to copy.
- You may drag within a given pane (moving or relating content on the diagram), within a window (dragging content from the palette onto the diagram), or between windows (moving content or relating elements).
- Pressing the escape key will abort a drag-drop operation.
- Dragging onto an element – whether a functional node on an EFFBD, a connecting link on an internal block diagram, or an element in a list – is always interpreted as a link operation to create a relationship between the CORE elements. You will be presented with a dialog of all valid

CORE 8 New Features Guide

relationships, and the key relationships within the semantics of the diagram are shown in priority order with a blank space separating these from all other valid relationships. For example, if you drag an item onto a function node in a sequence diagram, the triggers relationship will be shown first in the list, the outputs relationship second, a blank space, and then the inputs relationship (valid for the elements involved but not impacting the sequence diagram).

When using the New ABC options on the Constructs tab, you can create new CORE elements simply by dragging these constructs onto the diagram. In addition to dragging existing nodes onto the diagram, you can drag a New Node which will auto-create a new function and insert it accordingly. Some constructs (such as New Node) are intended to be dragged onto the diagram backdrop. Some are intended to be dragged on top of an existing diagram node (such as New Trigger). This can accelerate model development and enable you to move at think-speed, much like standing at a white board and sketching your system while having CORE build the model behind the scenes.

In the User Preferences under the Diagram >> General settings, there is a new option “Prompt for Element Names on Insertion”. By default, this option is disabled, so dragging a New ABC construct onto the diagram will create a new element with a default name. If you enable this option, when you drop a New ABC construct, you will be prompted to fill in the name before the element is created.

Tip: Sometimes, you want the best of both worlds – quick element creation for some drag-drop actions with the opportunity to fill out names for others. Holding down the CTRL key when dropping a New ABC construct onto a diagram will prompt you for the element name regardless of the User Preference setting.

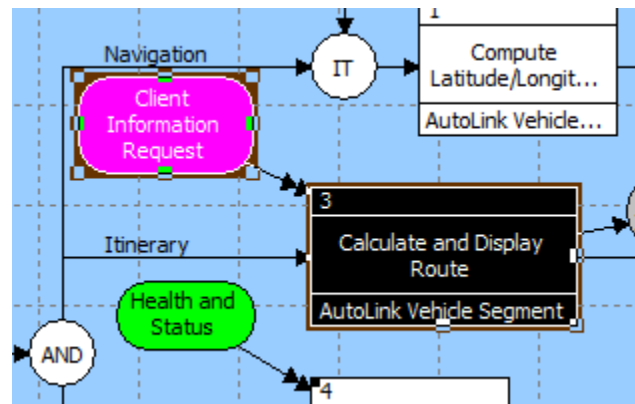
Moving and Resizing Diagram Content

The diagram framework has been revised to offer greater flexibility and usability when manipulating nodes. The result is a more powerful, more natural diagram to work with.

The first left-click on a node is the selection action. Clicking on a selected node (or nodes) displays a movement cursor allowing you to drag the selected nodes as desired. Some diagrams (block diagrams and package diagrams to name just two) are free-form and allow nodes to be moved anywhere. Some diagrams (sequence diagrams for example) have strict layout constraints with nodes constrained to move vertically along life lines. Other diagrams (EFFBDs and activity diagrams) represent a mix with constructs constrained to move horizontally along branches and overlay nodes (item and resource nodes) free to move anywhere on the diagram.

To resize a node, select the node and then adjust the desired sizing handle. Nodes indicate in which directions they can be resized. In the example below, the item icon has all 8 handles indicating it can be resized in any direction. The function icon only has right and bottom handle indicating its resize operations are limited to these aspects. Dragging a handle in the middle of a face constrains the node to resize in one direction only. Dragging a handle at a corner allows the node to resize in multiple dimensions simultaneously.

The notable exceptions to resizing are the N2 diagram and the IDEF0 diagram. Given the fixed layout of these diagrams, the nodes are not resizable. Also, on sequence diagrams, the component headers for lifelines and the corresponding function elements have fixed width. Finally, constructs are not resizable on FFBD, EFFBD, and activity diagrams.



Auto-sizing Icons

With the ability to manually resize individual icons on a diagram, you now have the ability to size specific nodes to display the needed content. Whether for a long function name on an activity diagram, a full description on a requirements diagram, or a picture of your system, you can now maximize the use of the available space to show the needed content.

CORE includes two different auto-sizing options that operate on both icons and labels:

- **Auto-size.** This command (available from the menu and the toolbar) adjusts both the width and the height to display the desired content. Starting with the icon width specified at the diagram level, the algorithm checks each fixed height field, increasing the width as required to display all content. Once the width is determined, the icon height is adjusted (up or down) until there is space to display the full content of all dynamically-sized fields. Note that the maximum icon dimension is still 500x500.
- **Auto-size Height.** This command (available from the menu or by holding down the control key while issuing the Auto-size command) keeps the icon width the same but adjusts the height to display the full content of all dynamically-sized fields.



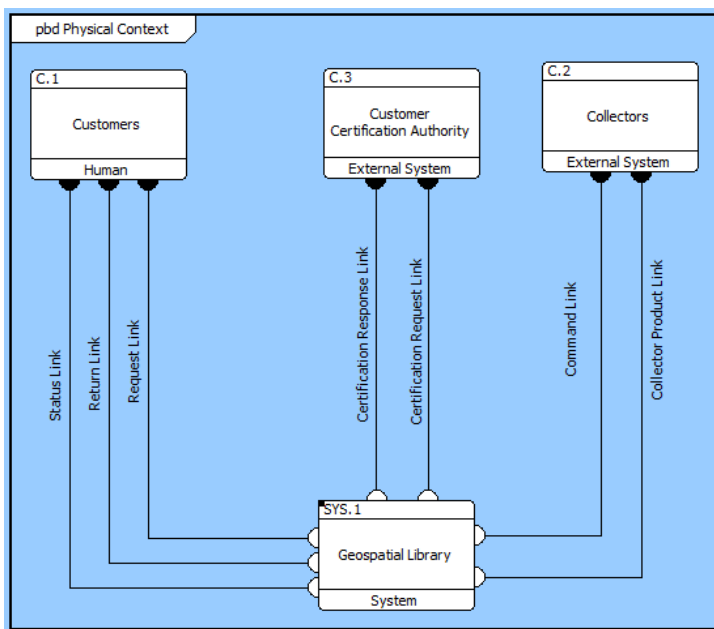
Both commands can be applied either to the selected objects (nodes or labels) or to the entire diagram. If no diagram objects are selected, you are prompted for confirmation before all diagram content is resized.

Enhancing Model Visualization and Communication

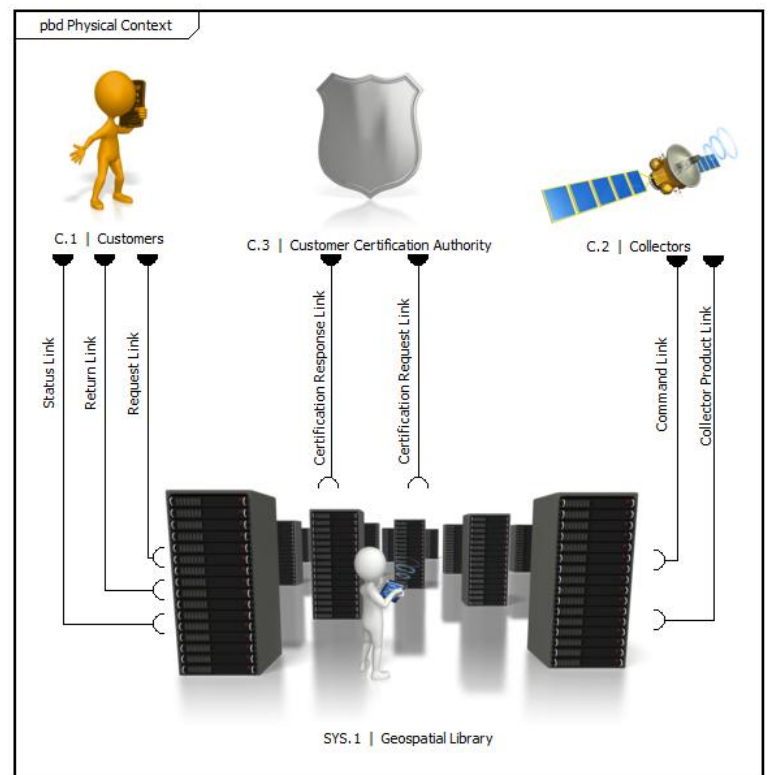
Traditionally, CORE has focused on technically correct diagrams. While some color and sizing flexibility existed, the emphasis was on the engineering, not the communication or presentation. CORE 8 represents a landmark shift, maintaining all of the power and consistency of the past and complementing it with robust and rich representations. With diagrams at the heart of the CORE and MBSE interactions to develop, manipulate, and communicate the underlying model, CORE 8 takes model-driven representations to a whole new level. We believe the result is a rich, flexible, and highly usable framework that delivers engineering analysis with style.

Graphical Images and Geometric Frames

CORE 8 introduces the ability to represent CORE elements as traditional geometric frames or as graphical images. For an individual node or the entire diagram, you can select the representation that best suits your needs. In this case, a picture is truly worth a thousand words.



A Traditional CORE Diagram



The Same CORE Diagram using Graphical Images

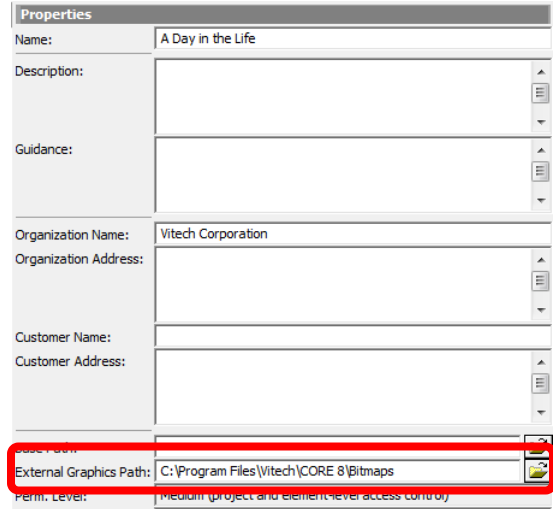
Behind this feature are a host of changes that provide a rich and flexible framework. Following is a step-by-step walk-through of the various aspects from the project-level on through the individual diagram node.

External Graphics Path

CORE 8 includes a base library of over 3,000 images to draw upon. That said, we expect that many times users will want to use their specific image on a diagram. As with the existing external references in CORE, user-provided graphical images used on diagrams are maintained and managed externally to CORE. This provides a highly flexible, high performance system allowing you to reference the graphic of interest. To simplify management of these external graphic files, each project has a setting for its base graphics path.

By default, the external graphics path is set to the Bitmaps folder in the CORE installation directory. You can add any graphic desired to this folder or can change the path to point to any directory you wish. Note that for team configurations, you will most commonly want to use a common network location.

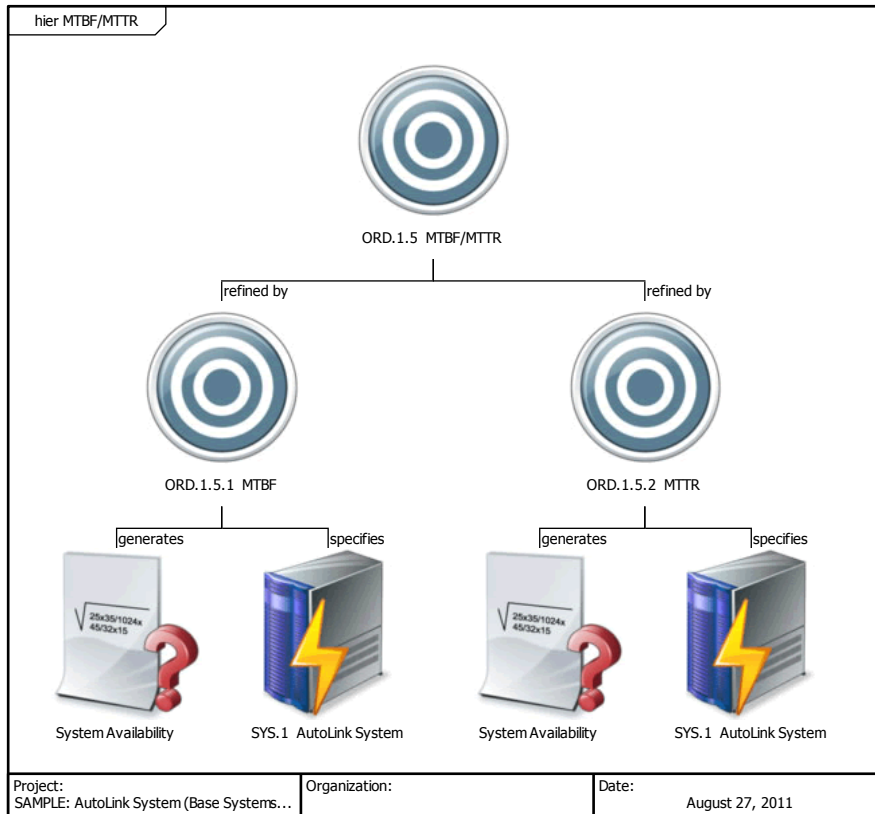
Analogous to other CORE external references, individual images can be located either within this base graphics path or outside of it. Images within this directory structure will automatically be referenced relatively, easing the management from machine to machine. Images outside of this directory structure will be referenced absolutely, meaning that multiple users must be able to reach the same location using the same network path.



Assigning Images to Database Classes

CORE 8 includes the ability to associate images with database classes. Not only does this enable the display of graphical images on the schema-side diagrams, it provides a default image for all elements of that class. As with the color model in CORE, when elements are queried for their graphical image, they first look to see if an image has been specified directly on the element. If not, they then look to their class and return the image there, if any.

All CORE schemas have been extended with base representations for each class. This provides a strong default representation – without any special preparation on your part – the first time you toggle nodes to their image representation.



Associating Images with Elements

CORE 8 New Features Guide

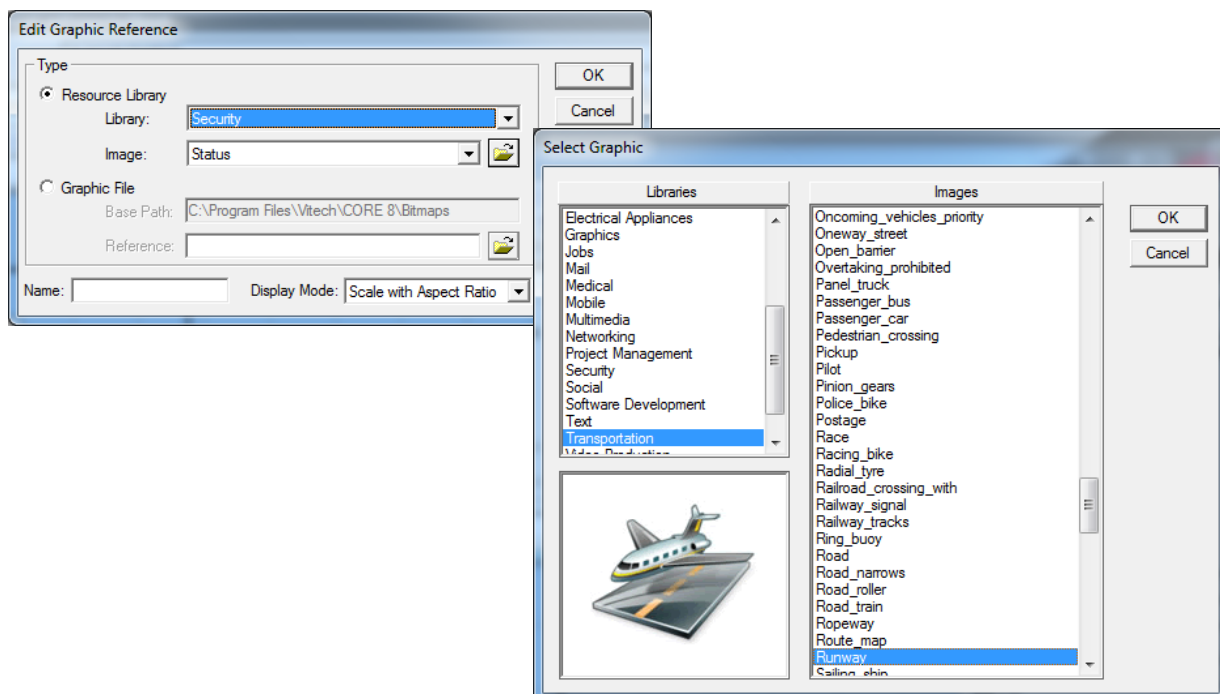
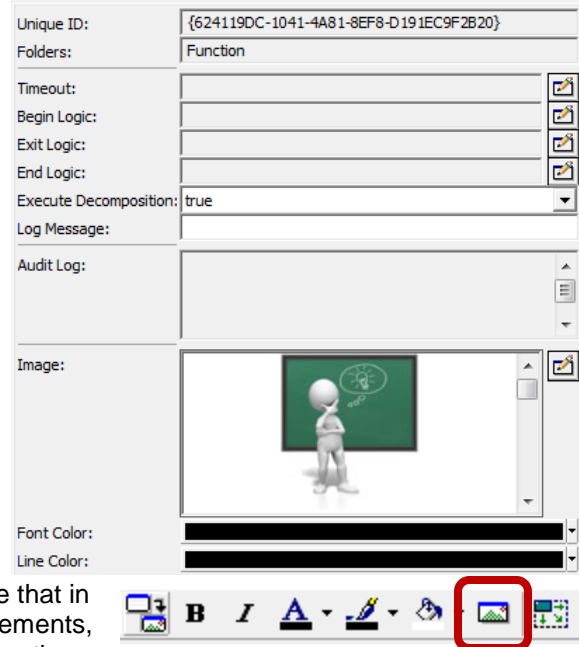
While images can be (and should be) used to visually represent database classes, the primary use will be to associate images with individual elements. All CORE schemas have been extended to include a new bitmap attribute (aliased as “Image”) of type graphic for all classes. This allows a stock image from CORE’s library or a user-provided image to visually represent an element on any diagram. This is particularly useful for components and functions.

By default, the bitmap attribute is shown on the secondary tab of the property sheet, immediately above the element color profile. The property sheet displays the current image identified for the element, and the edit button in the upper right of the pane allows you to identify the appropriate external image.

In addition, a command to set the bitmap for the selected nodes has been added to diagrams. This eliminates the need to open the property sheet, select the second tab, and then invoke the image selector. Note that in addition to setting the bitmap attribute for the selected elements, this command toggles the node to use the image display rather than the geometric frame.

As noted, CORE 8 includes a base image library of over 3,000 images organized into approximately 25 categories to draw upon. All of the images included are 128x128, well-sized to balance representation of the image with the overall size (in pixels) of the diagram. While we expect that you will want to use custom graphics, providing a common library significantly lowers the barrier of entry.

When specifying an image – either to associate with an element, a class, or a free-floating graphic (more on this later) – you will see a dialog that allows you to either select from a stock library or to navigate to a custom file of your own. If you know the stock image you want, you can directly select the desired library and image from the drop-downs. More likely, you will click the folder icon to browse the available library.



In addition to selecting the category / image from the stock library or specifying your own file, there are two additional aspects associated with images:

- **Display Name.** An optional string provides a flexible, user-assigned name for the external graphic reference. This name is used to represent the graphic reference in table cells and reports.
- **Display Mode.** The scaling approach to be used when this image is displayed on a diagram. There are three possible values:
 - *Maintain Fixed Size.* Display the image at its native resolution, cropping the image as required. This mode will not scale either dimension up or down.
 - *Scale with Aspect Ratio.* Scale both dimensions proportionally to fit the available space. Note that the maximum scale in this mode is 100% in order to maximize the impact and usability of the diagrams. (This is the default value. Note that regardless of which mode is specified for diagrams, the property sheet shows the image in fixed size mode.)
 - *Stretch to Fit.* Scale both dimensions independently up or down to fit the available space. The image will stretch or compress as required.

Tip #1: Where stock images fit the need (technical and presentation) of the project team, selecting a stock image significantly simplifies model management. Rather than having to maintain a separate library of image files, everything is self-contained in the graphic library installed with CORE. However, more often than not, we expect you will want your own graphics to convey either the richness of the domain or your specific branding.

Tip #2: Don't overdo a good thing. Emphasize graphics for level 0 / level 1 and critical stakeholder communications.

Tip #3: Use moderation in your color palette. CORE supports up to 16M colors, but 256 color bitmaps produce almost the same level of results.

Tip #4: Use moderation in image size. A 128x128 image gives richness without increasing the diagram (pixel) size.

Tip #5: When toggling between geometric frames and graphical images, the auto-size command is your friend. You can quickly resize a specific icon or – if nothing is selected – resize all nodes on the diagram.

Specifying Whether to Use Graphical Images or Geometric Frames

As with color profiles, CORE provides the ability to specify whether to use graphical images or geometric frames at both the diagram level (via the defaults and the options) and at the individual node level. Within the diagram preferences, each icon profile includes a button to toggle between geometric frames and images alongside the font style.

Using the presentation menu and on the toolbar, this diagram level setting can be overridden and the display toggled for the selected nodes. The graphical display option is available for all single element nodes. This applies equally to physical block diagrams and block definition diagrams, EFFBDs and sequence diagrams. The option is not available for connections (where elements are represented by lines and arrows such as links on a physical block diagram or triggers on a sequence diagram). This option is also not available for multi-object nodes such as classes on an ER diagram or items on an N2.

When a node is set to display with a graphical image, the specified icon template is used to display a label centered below the image. The individual fields are concatenated into a single string with separators translated to a vertical bar (|).

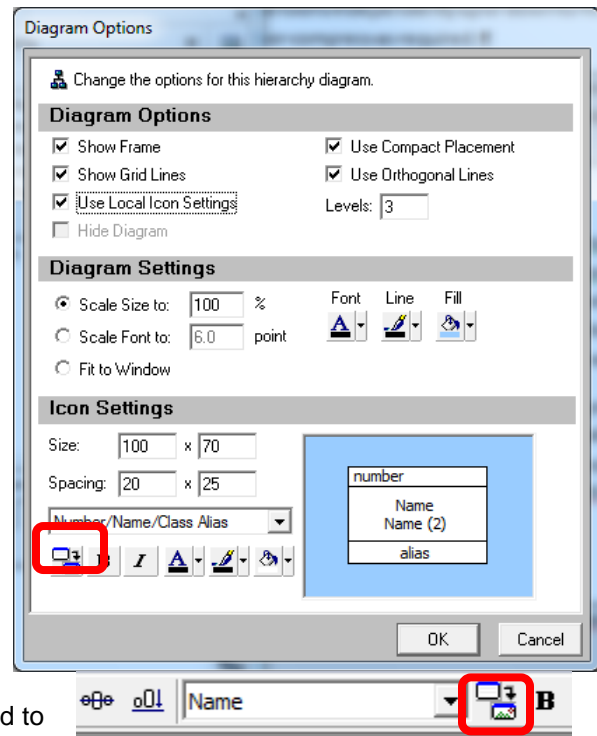
In allocating the space available on an icon to displaying a graphical image, first priority is given to the image itself.

- If no image has been specified (or if the specified image is not accessible), no image will be displayed.
- If the display mode is fixed, the image region will scale up until the full image is displayed. The remaining space will then be used for the label.
- If the display mode is maintain aspect ratio or stretch to fit, the image region will first scale up until the scale reaches 100%. The remaining space is allocated until the entire label can be displayed. Once the full label is accommodated, any additional space allows the image to display at more than 100% scale (for stretch to fit).
- At least one line is reserved for the label.

While the line color is not used in displaying a graphical image, the font color is used for the label, and the fill color is used for the node background (visible behind the label).

Tip #1: When using graphical images, it is often best to switch from CORE's classic pale blue background to plain white.

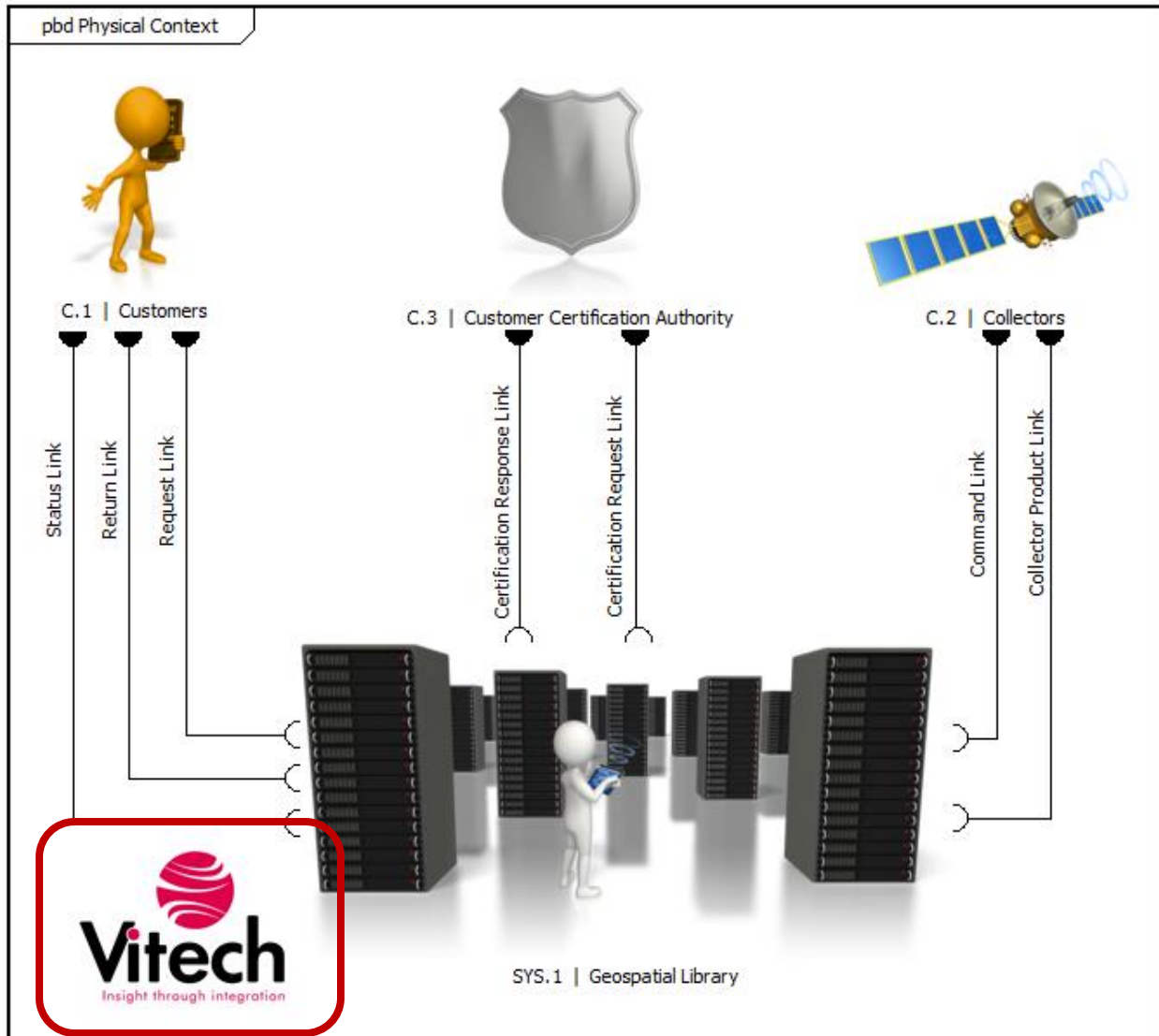
Tip #2: While you can use any icon template in combination with a graphical image, classically less is more. A simple number/name (without a separator) or just the name may look better than a more complex representation.



Inserting Generic Graphics

With the introduction of image representations for elements, the natural next question is “can I put a picture of X on my diagram?” There are many useful cases for this, most notably team or project logos as well as graphical backgrounds. To support this need, CORE 8 includes the ability to insert an independent graphic on a diagram.

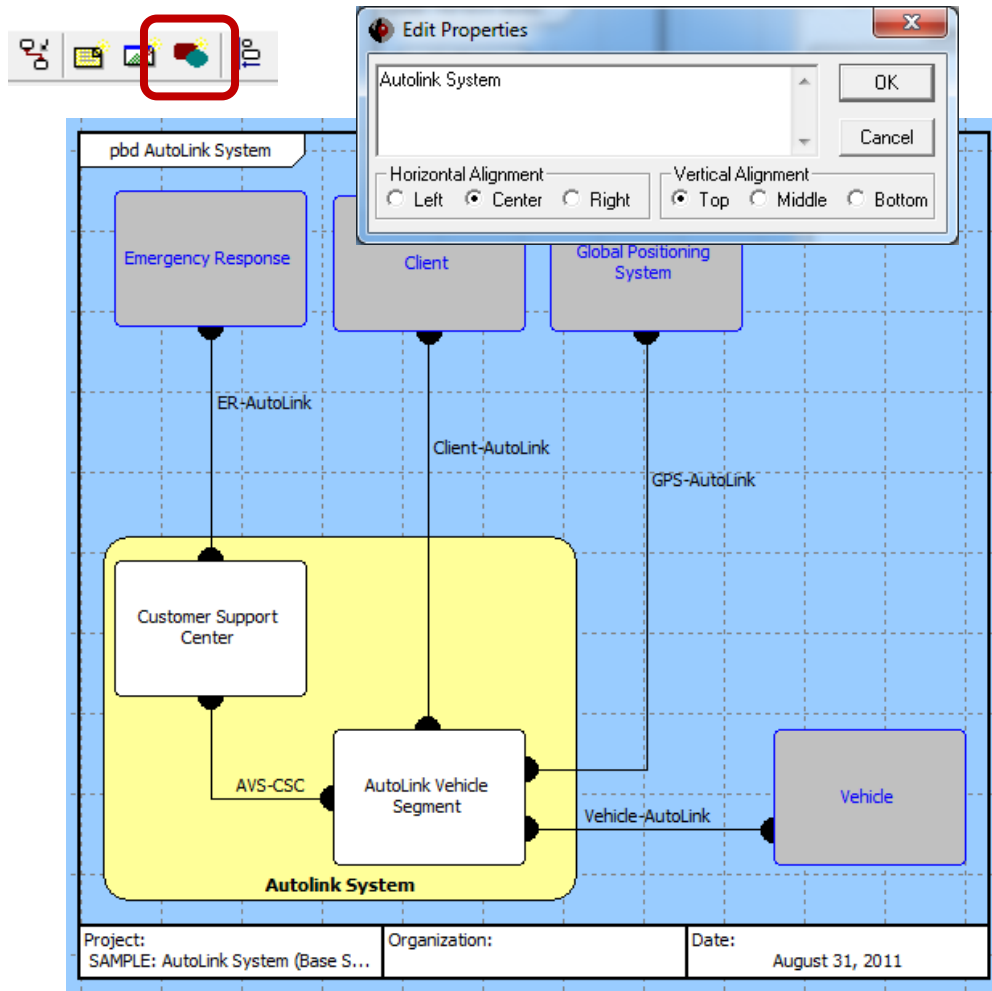
Shown on the toolbar adjacent to the Insert Note command, the Insert Graphic command prompts you to select an image from either the stock image libraries or from your own file collection. The image is then inserted at full size in the middle of the screen. You can then easily drag the graphic to the position desired.



Inserting Generic Shapes

Complementing the existing CORE 7 capability to insert notes and the new capability to insert graphics, CORE 8 introduces the ability to insert generic shapes on diagrams. Rectangles, rounded rectangles, ellipses, and circles are all supported. Like notes, diagram shapes also can optionally include text. This enhances the representation and communication value by emphasizing key aspects or implying groups, clusters, etc.

When selecting the Insert Graphic command (most likely, using the toolbar button), you will be prompted to frame the region for their shape. You are then prompted with a dialog to specify any text and its positioning within the shape (centered at the top by default).



Font Decorations

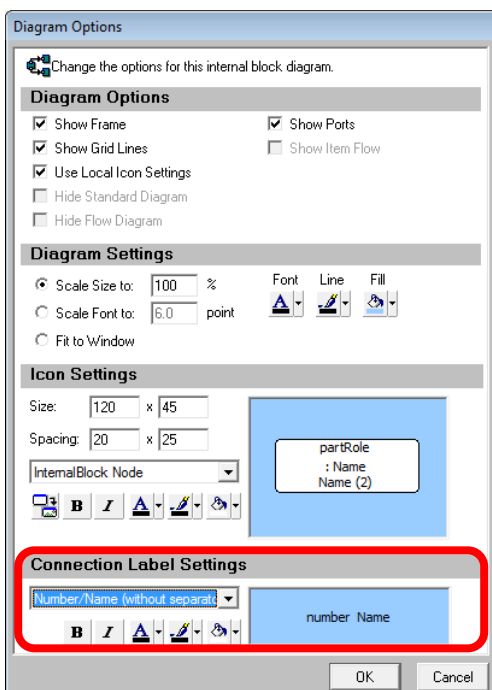
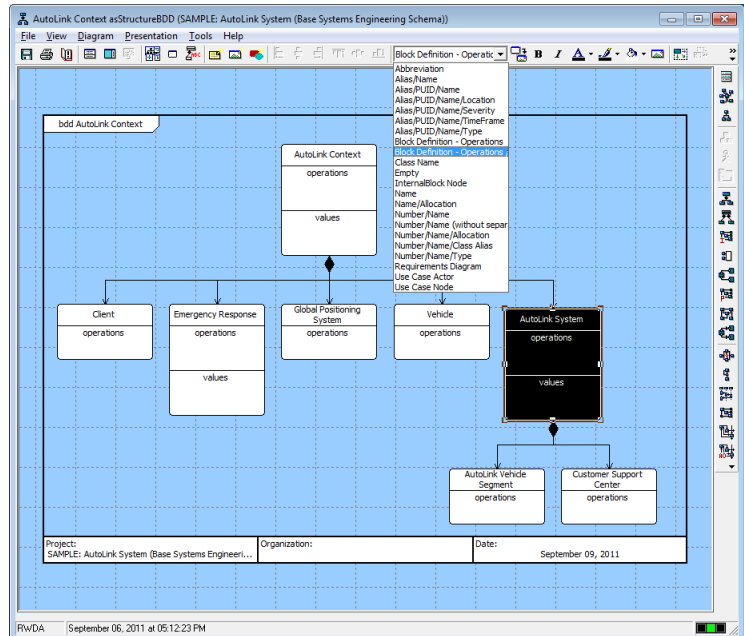
Previously, CORE diagrams were limited to a single font presented in normal format. While certain conditions (e.g., an unallocated lifeline in a sequence diagram) made use of formatted text, you had no ability to control the font. With CORE 8, you now have the ability to bold and italicize text at the individual node / line level. Complementary to setting font, line, and fill colors, font decorations help call attention to specific diagram content.

Icon Templates

Historically accessed via the User Preferences, icon templates provide you the ability to control how objects are represented on diagrams. While we have made numerous improvements over the years (adding labels, enabling the display of relationships, etc.), most users may be aware of the ability to set icon sizes, but they are not aware of the ability to set the icon template for a diagram. More than that, they are not aware of their ability to define their own specific icon templates.

In CORE 8, we have both increased the power of icon templates and their visibility / usability. First, rather than per-user customization, icon templates are now defined at a project level. This allows all team members to leverage a common library. Furthermore, icon templates are now accessible in the Utilities section of the project explorer list rather than being buried in the User Preferences.

Previously, icon templates were defined per object type, per diagram. In other words, you could have one icon template for functions on an FFBD and another icon template for the items. Now, you can also assign an icon type for an individual icon. So if you have a block definition diagram where you want to show operations for one node and values for another, you simply select the desired template from the drop-down menu in the toolbar.



Icon templates can also now be associated with connections on diagrams as well as with icons. For example, on a physical block diagram, you can specify to display not only the link connecting two components but also the items it carries. On a sequence diagram or an IDEF0, you can decide to show abbreviations rather than full item names. For each of these diagrams, the diagram options have been extended to include a connection icon template in addition to the existing node icon templates. Of course, you can also select an individual connection and change its icon. Again, it's all about providing flexibility and richness to better enable representation and communication.

Set as Default Icon

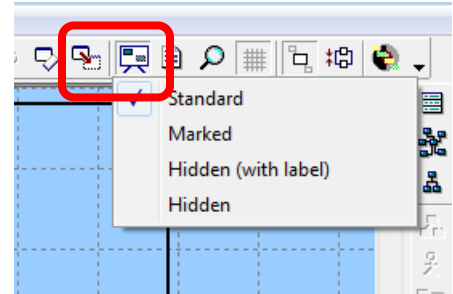
Traditionally, users had the ability to set the parameters for the default diagram icon from the diagram options dialog. As more direct manipulation – of icon size, color, font, and template – is done directly on the diagram, it becomes natural to customize an icon on the diagram and then set it as the default for the

icon. Modeled after the PowerPoint® capability, CORE now has a Set as Default Icon command that does exactly that. The parameters can still be set via the diagram options dialog, but this immediate command allows greater direct manipulation and diagram customization.

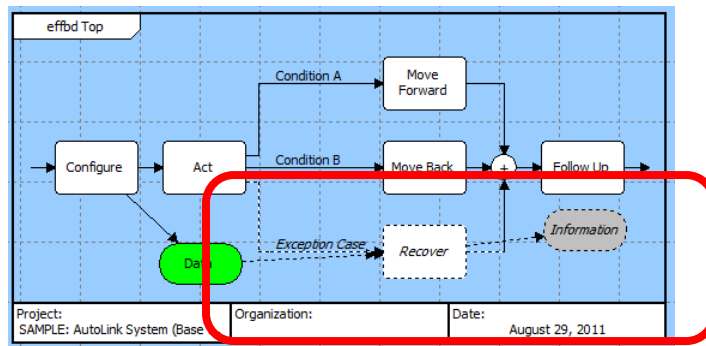
Hiding Diagram Content (Eliding in SysML Terminology)

CORE has always excelled at presenting technically correct and technically complete graphic representations. Unfortunately, comprehensive and complete diagrams can complicate communication. CORE 8 introduces the SysML concept of eliding content, more commonly known as selectively hiding / showing content. This capability allows CORE to deliver comprehensive, complete, correct representations from the underlying model without sacrificing communication. This enables you to focus audiences on the areas of interest without having to manage and synchronize separate artifacts.

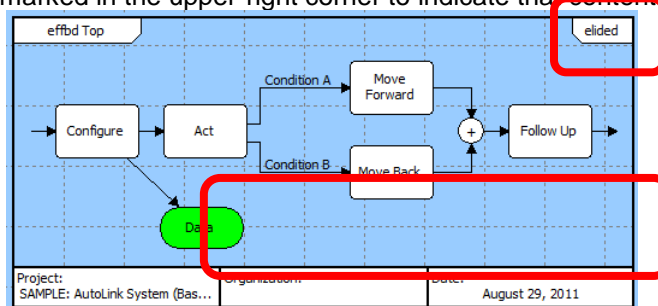
Individual diagram objects can be selected and marked to hide / show using the command in the toolbar. Diagrams now support four specific display modes to leverage this capability.



- Standard.** This display mode maps to the traditional CORE view. All diagram content is shown. No visible markings indicate any diagram objects that have been marked to hide. The standard view is most appropriate for engineers and analysts as they architect and define the system.
- Marked.** To help manage which objects are marked to be hidden, the Marked display mode displays all diagram content but shows content that will be hidden with dotted lines. When leveraging the show/hide capability, this mode is essential to managing diagram appearance.



- Hidden (with Label).** In this mode, diagram content marked to hide is hidden from view. This could be individual nodes, constructs, or entire branches. Layouts are compacted such that there is no layout cue that content is missing. In fact, dependent objects (for example, items on an EFFBD that are not connected elsewhere on the diagram) are also automatically hidden. The diagram frame is marked in the upper-right corner to indicate that content has been hidden.

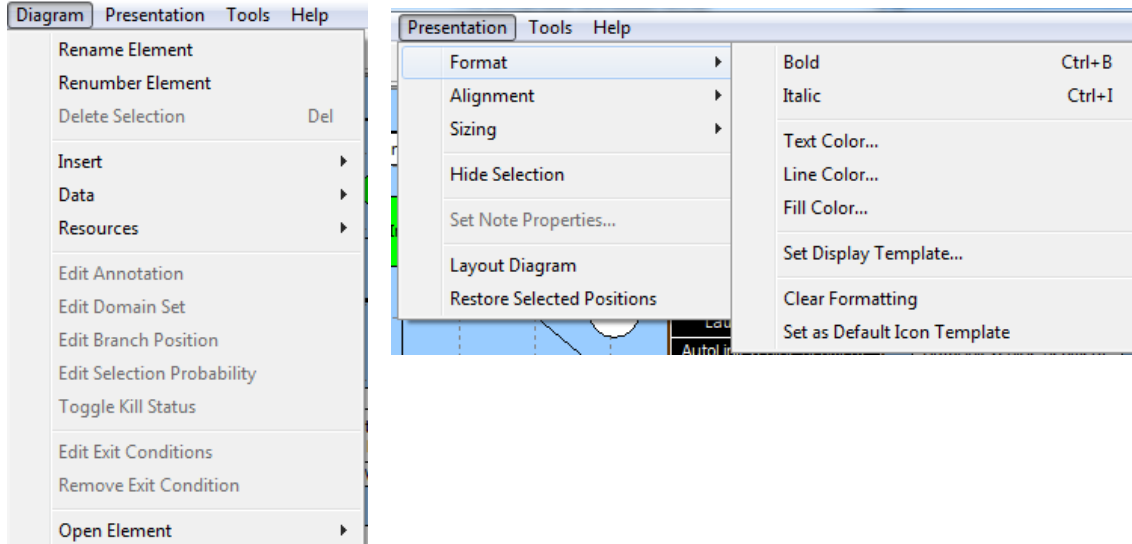


- Hidden.** This mode is the same as the Hidden (with Label) mode but without the diagram frame marking. This mode is most appropriate for use with customers so that they can focus on the diagram content (rather than thinking about what has been hidden).

Note: Eliding one node on a diagram can also elide dependent content. For example, if you elide a function on an activity diagram, any data that is not related to a function marked to show will be elided. Likewise, on a block diagram, if a node is marked as elided, all corresponding connections will also be elided. On the IDEF0 diagram, each individual object must be elided as desired. This provides the base capability but without the degree of intelligence / automation that exists on the other diagrams.

Presentation Menu

Given the notable changes in the formatting options on diagrams, we have taken this opportunity to restructure the diagram menus into two menus – a Diagram menu for commands that manipulate the diagram content and a Presentation menu for commands that manipulate the diagram representation.



Additional Diagram Framework Changes

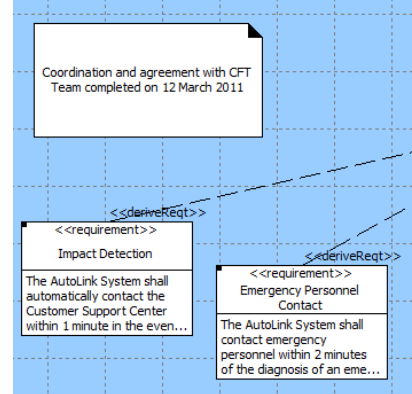
A few additional bits and pieces have been added to the generic diagram framework to add that final polish – both in presentation and in usability.

- With the exception of the primary diagonal on an N2 diagram, all geometric frames now have a slight rounding to the corners. This modernizes the appearance without creating confusion between the classes of information on icons.
- With all of the formatting options available, CORE 8 has added an option to clear formatting for the selected nodes. This command resets the node to the default presentation specified in the diagram options – the icon size, font decoration, icon template, etc.
- While the Clear Formatting command allows you to return a specific node to the diagram defaults, sometimes this is akin to killing a fly with a sledgehammer. Perhaps the most notable case is attempting to restore the icon size to its default value (particularly after switching between various representations). To address this case, a Reset Icon Size command has been added to the Presentation>>Sizing menu.
- A Select All command (Ctrl+A) has been added to the Diagram menu. This makes it convenient to select all nodes and perform a graphic change (toggle a status, resize, etc.). Note that the utility of this command on EFFBDs, activity diagrams, and sequence diagrams is somewhat reduced given the construct selection behavior.
- The color controls in a diagram toolbar now apply to the diagram itself if no objects are selected. This makes it easier to switch the diagram background color to white when using images instead of geometric frames.



CORE 8 New Features Guide

- The grid lines on diagrams are now constant in their position rather than “drifting” as the diagram grows. Holding a consistent position relative to the diagram frame (rather than the window corner) makes the gridlines much more useful for alignment and spacing purposes.
- The Restore All Base Positions command has been renamed Layout Diagram to adopt standard terminology.
- Diagram notes have been revised to use a more traditional “note” shape. The options have been reduced to be more consistent with other diagram objects. Also, a default color template has been added to control the default presentation (light yellow by default).
- Within the User Preferences, the general diagram settings (controlling the frame block and construct colors) is now on a new General Diagram tree item to increase visibility.
- While the FFBD manipulation toolbar (with all of the constructs) is still defined, it has been toggled off by default. This toolbar is largely redundant with the introduction of drag-drop from the palette.
- The option to auto-hide extension points on the use case icon template has been removed. With the addition of per-object icon templates, this option is no longer required.



Expanding Our Toolkit of Representations

Systems engineering is challenging. With diverse analytical and communication demands, we need a rich and diverse collection of integrated representations at our disposal. CORE 8 continues to add to that collection.

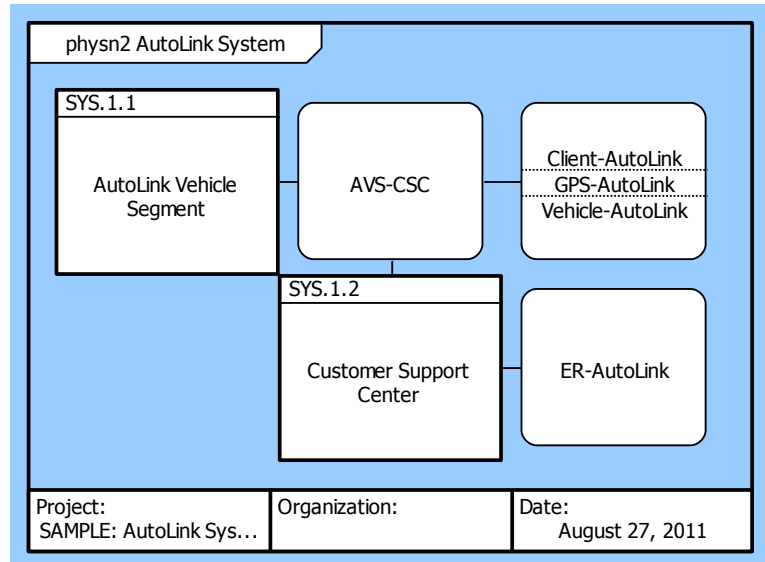
Interface and Physical N2 Diagrams

With the recent explosion of physical (and behavioral) representations available in CORE, there remained a high-level void that is now served by interface and physical N2 diagrams.

These diagrams complement the existing block diagrams, presenting the information in a more structured manner suitable for interfacing, clustering, and packaging analysis.

Both diagrams use the traditional N2 representation of primary blocks on the diagonal and related objects on the off-diagonal. In these representations, the child components are shown on the diagonal. The off-diagonal represents the interfaces (interface N2) or links (physical N2) that connect the components.

In this format, the nature of the connections becomes clear. You can quickly assess whether the model has a single highly connected component (which could represent a technical or interface risk), multiple clusters of interconnected components (valuable insight for packaging), etc. While the same information can be gleaned from a block diagram, this insight screams out when presented in an N2 format.



There are several notable items with respect to physical representations of N2 diagrams:

- Whereas there is clear directionality in a behavioral N2 (an item is output from one function and input to or triggers another function), the physical N2s speak more to connection than directionality. For that reason, no arrows are shown on the physical N2s. Instead, the diagram simply represents who is connected to whom.
- The lack of directionality means that half of the off-diagonal locations are redundant. If A is connected to B, we know that B is connected to A. Rather than showing this information twice, only the upper half of the diagram is used. The lower off-diagonal cells will be empty by definition. In addition, the external input and external output locations become redundant. For this reason, we have hidden the external input cells at the top of the diagram.
- The physical N2 diagrams are generally higher level diagrams. For this reason, they are level 0 diagrams which do not use the connected thru and joins thru relationships. Instead, they rely solely on the connections defined for the specified components.
- External connections on a physical N2 represent one of two conditions. First, it may be a connection with an outside component or part of the environment. Second, it may be a connection with a child of a component not shown on this diagram. In that case, the connection should be with the component itself, not the child. This connection can then be further elaborated at a lower level of the physical architecture.

There is one notable command that has been added to the N2 diagram to support clustering analysis for both physical and behavioral N2 diagrams. The Diagram>>Change Node Position allows you to change the position of the selected component / function on the diagonal. The diagram then redraws the matrix in

this configuration. This capability dramatically increases the value of the N2 in visualizing (behavioral) allocation and (physical) packaging.

Note: By default, the physical N2 diagrams compute their order from the sorted order of their children. The behavioral N2 diagram computes its order by traversing the decomposition structure (visible in an activity diagram or an EFFBD). Once you manually change the node position on an N2 diagram, all future additions will be appended to the end of the diagonal. Resetting the diagram layout via the Layout Diagram command will restore the computed order, and CORE will once again maintain the order automatically.

Spider Diagrams

Traditionally, CORE has focused on structured engineering representations with defined formats – EFFBDs, hierarchies, sequence diagrams. These formats are critical to the engineering analysis but sometimes limit greater insight into and communication of the essence of the system. The spider diagram helps restore and highlight that context.

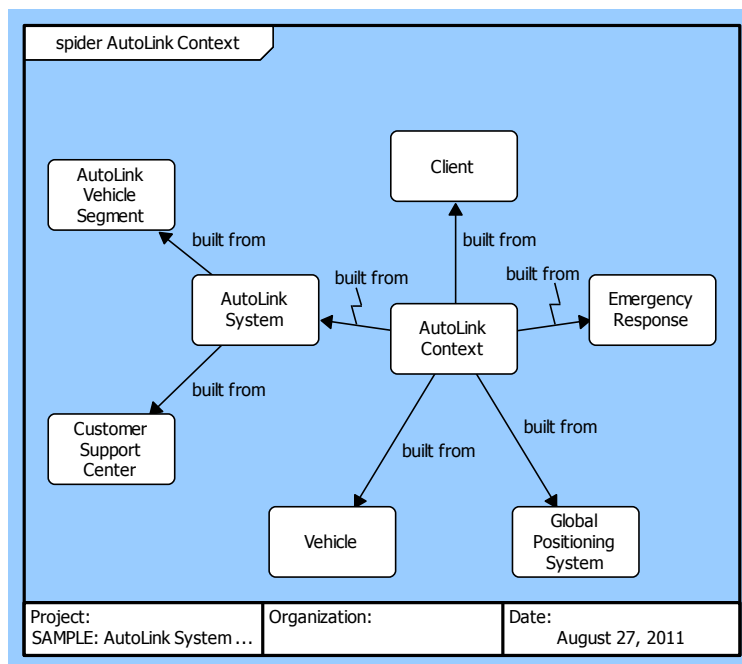
Presented in a free-form layout, the spider diagram presents a complete contextual view of a set of entities and their interrelationships. The diagram blends concepts present in an ER diagram (displaying relationships relative to an entity of interest) with concepts of a hierarchy diagram (showing multiple levels of relationships). However, unlike a hierarchy diagram, each entity is represented once and only once. In addition, the free-form presentation does not artificially imply a hierarchical relationship that may not exist. The result is an extremely powerful representation – neither traditional nor SysML – to further enrich our toolkit for analysis and communication.

Element Spider Diagrams (aka Spider Diagrams)

Available on all elements regardless of class, the spider diagram is opened on a specific element and hierarchy definition. The element represents the center of reference for this diagram. The hierarchy definition specifies the relationships and target classes of interest.

From a command and preferences perspective, the spider diagram has a great deal in common with the hierarchy diagram:

- The diagram is opened to a default number of levels (specified in the preferences). You can change this number in the diagram options or expand / collapse specific nodes of interest. Expanding and collapsing is handled either by the + / - commands in the toolbar or by control-double-clicking on the nodes of interest.
- The diagram is defined by the combination of a central entity and a hierarchy definition. You specify which hierarchy definition is desired when opening the diagram (the default associated with the class is used on the project explorer tab). A command to switch to a different hierarchy definition is available from the toolbar. Like the hierarchy diagram, because each entity can have a number of different spider diagrams associated with it, CORE automatically tracks and reuses the settings associate with each specific combination of element and hierarchy definition.



switch to a different hierarchy definition is available from the toolbar. Like the hierarchy diagram, because each entity can have a number of different spider diagrams associated with it, CORE automatically tracks and reuses the settings associate with each specific combination of element and hierarchy definition.

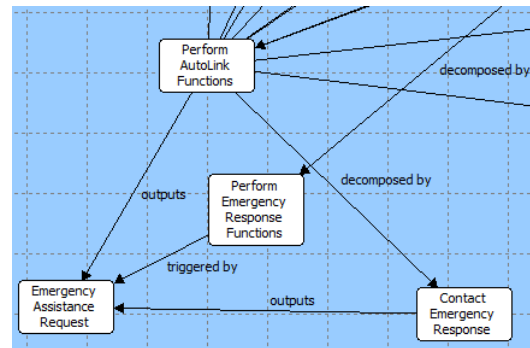
- An additional hierarchy definition (All Relationships) has been added to present a full context view of the selected element. This hierarchy definition is auto-maintained to

include every relation in the schema. Therefore, it is complete. However, it should be used with care (open a diagram using this hierarchy definition and you will understand why this is called a spider diagram).

- The commands to remove / delete elements are available from the Diagram menu. Adding elements is best handled by dragging elements onto nodes from the diagram palette to establish the desired relationships.

While there is a great deal of commonality (and one could argue repetition) between the hierarchy and spider diagram, there are a number of notable differences as well.

- The spider diagram is a free-form layout. Nodes can be moved anywhere on the diagram, and labels can be moved as well. This presents great flexibility in presentation, but also transfers additional maintenance to you (with great power comes greater responsibility). In this aspect, the spider diagram has more in common with the block diagrams than anything else.
- Whereas the same element can be repeated on a hierarchy diagram (with only the first occurrence being expandable), an entity will only appear once on a spider diagram. This emphasizes the interconnected nature of the model.
 - Note that this representation introduces some key interaction differences. If you select a node on a spider diagram and ask to remove it, you will remove all relationships which connect it to elements on the diagram. If you wish to remove just a single relationship, select the associated label and use the Remove Target command.



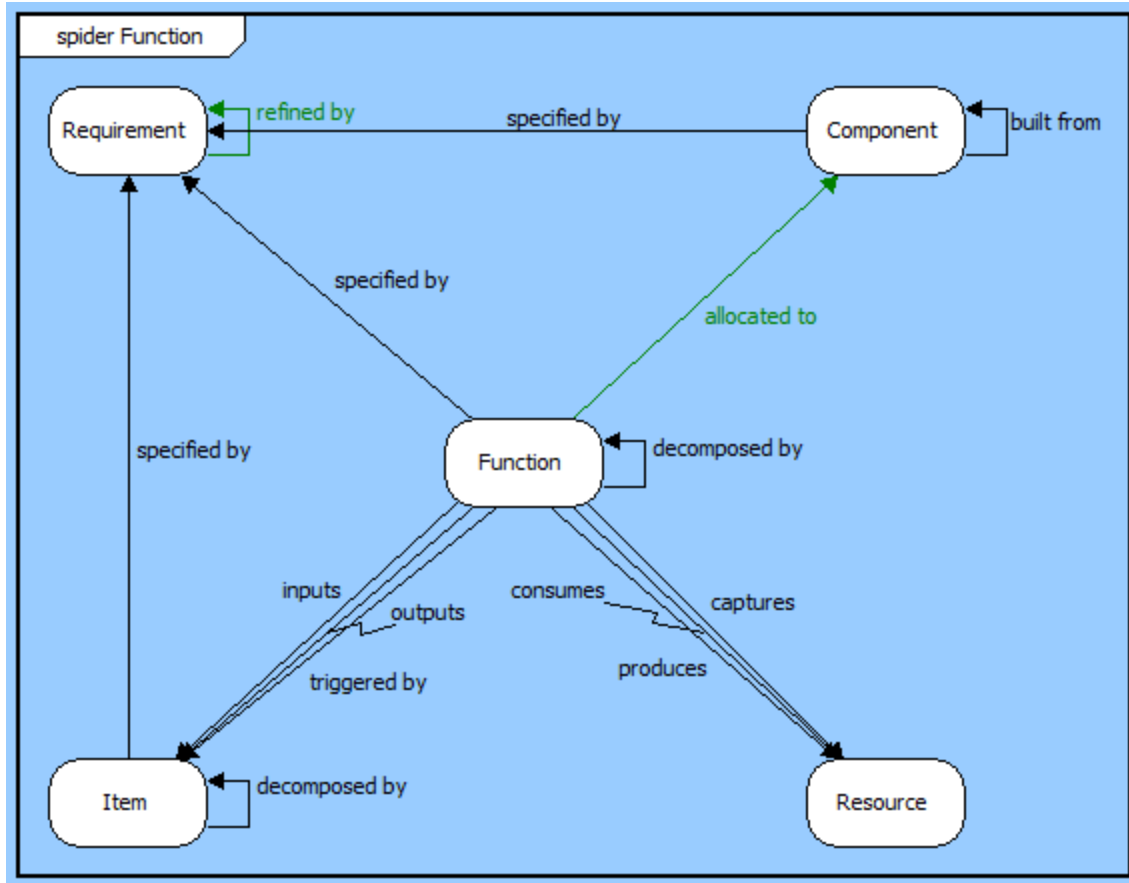
In addition, color profiles (font, line, and fill colors) have been added to relationship definitions, particularly to support the spider diagrams. By default, the font and line colors for traceability hierarchy relations (allocated to, basis of, causes, documents, generates, refined by, results in, and specifies) have been set to green. This color change helps highlight specific relations of interest (in the case of the default relations, this helps differentiate primary traceability).

Tip: Between its free-form nature and the emphasis on communicating / emphasizing certain connections, the spider diagram is a prime candidate for using the ability to hide specific nodes. Hiding the node will hide all relationships that connect to the node, allowing the audience to focus on other aspects of interest.

Class Spider Diagrams

In order to provide a roadmap to the MBSE language, a schema-side variant of the spider diagram has been developed. Opened on the combination of a class and a hierarchy definition, the class spider diagram presents a contextual view of how a type of element fits into the overall system model. Like other schema-side graphical representations, the class spider diagram is representation-only. The presentation of the diagram can be changed (nodes expanded, labels moved), but manipulation of the underlying schema is still managed through the project explorer.

Tip: When displaying a class spider diagram, it may be more valuable to specify the target classes of interest than the relationships. By doing so, you specifically identify the nodes that you want shown on the diagram. This is often easier and more valuable than identifying the connections (the relations). To do this, add all relationships to the hierarchy and then only select the desired classes from the hierarchy definition dialog.



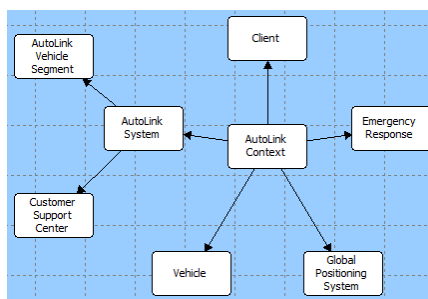
Layouts

With the introduction of spider diagrams, we have also introduced the ability to select from multiple layout formats for a given diagram. Sometimes, data will present better in one layout or another, either given the nature of the data, the intended communication/message, or the audience. Set as a default in the diagram preferences, the desired layout can also be changed on the fly via the layout menu.

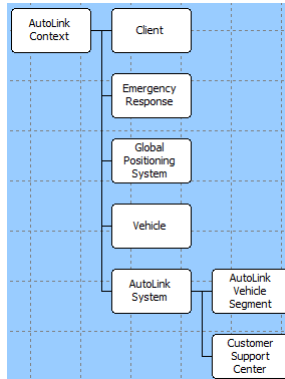
Note: The layout feature has been added to hierarchy, requirement, and block definition diagrams as well. The introduction of these layouts breathes new life into the hierarchies in particular. In some cases, a different layout fundamentally changes the usability and presentation of the data.

Currently, there are eight layouts available:

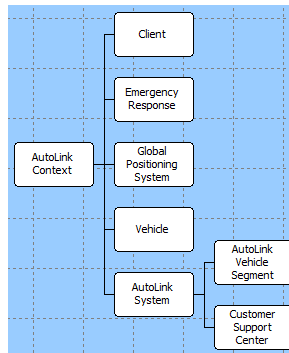
- Circular.** The default layout for a spider diagram, the circular layout organizes each level in a concentric circle. This representation is useful for highly interconnected data sets but can quickly turn into string art. *(This layout is not available for hierarchy, requirement, or block definition diagrams.)*



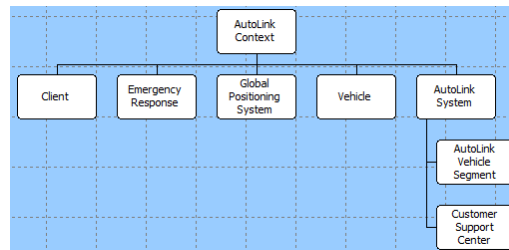
- Hanging.** The hanging layout is a lateral representation where child nodes “drop down” from the parent’s level. The result is a very clean representation, but one which can consume a great deal of vertical space.



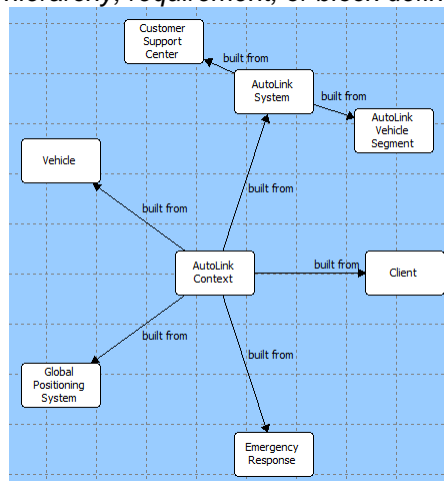
- Horizontal.** The horizontal layout takes the traditional vertical layout and simply turns it on its side. The child nodes are balanced vertically above and below the parent node. Given the content of requirements diagrams, the horizontal layout is often a good option.



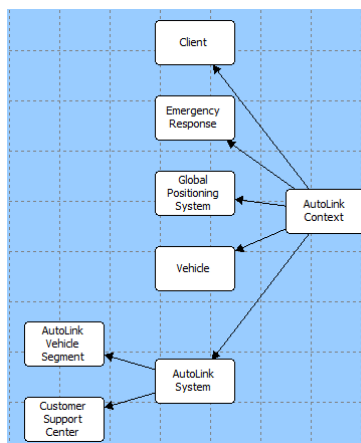
- Organization Chart.** Perhaps the most space efficient (and modern looking) layout for data sets with three or four levels, the organization chart displays the root element at the top, then displays the next level in a traditional vertical representation, and then organizes lower levels in a hanging format off of their parent. Though labeled organization chart, this representation is appropriate for many different data sets.



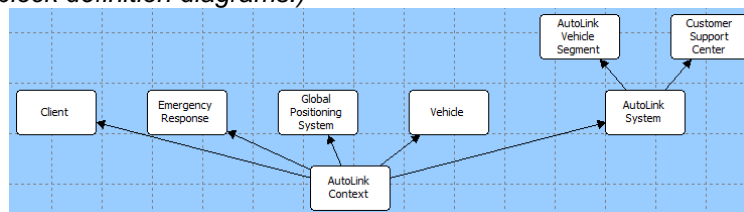
- **Radial.** The radial layout uses a hub and spoke model emanating out from the root node. *(This layout is not available for hierarchy, requirement, or block definition diagrams.)*



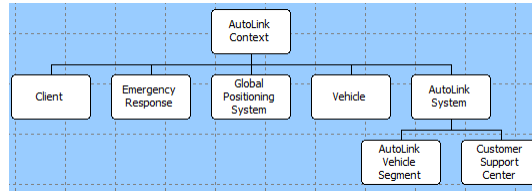
- **Right to Left.** The right to left layout is a variant of the Horizontal layout that simply reverses the direction, expanding the tree from right to left. For some spider diagrams, the implied semantic may work better than the “expanding” feel of a Horizontal layout. *(This layout is not available for hierarchy, requirement, or block definition diagrams.)*



- **Upward Tree.** The upward tree layout begins at the bottom and then spreads outward as it grows upward (the reverse of the Vertical layout which flows downward). For some cases, the upward growth may better match the desired semantic. *(This layout is not available for hierarchy, requirement, or block definition diagrams.)*

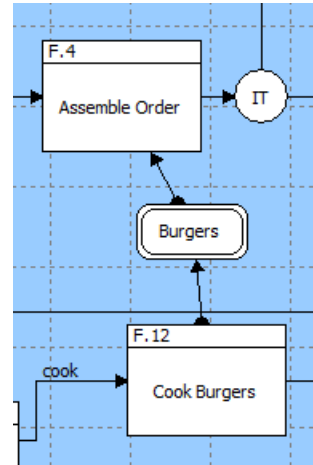


- **Vertical.** The default layout for a hierarchy diagram, the vertical layout is often referred to as the “standard” layout. It’s a classic representation for hierarchies and organization charts. The visual presentation often implies a flow down.



Additional Representation Changes

- Resources are now optionally displayed on EFFBDs. Resources are drawn with a double border. Consumes and produces relationships are indicated with a half circle and arrowhead decoration to reflect the direction of the flow. Captures relationships are shown with arrowheads on both ends. Resources have long been the behavioral aspect hidden from the behavioral views. With this revision, the EFFBD now provides a comprehensive visual of the elements and relationships that influence behavior. (Note that resources are not shown on activity diagrams as they are not part of the SysML definitions.)
- The database view menus and toolbars have been reorganized along logical lines (all interface diagrams together, all link diagrams together) vs. diagram lines (all block diagrams together, all SysML block diagrams together). This also better mixes the SysML and traditional views, reinforcing our blended representation philosophy so that you have the maximum number of representations available for the task at hand.
- The requirements diagram has been extended to open on a document element. While this does not strictly comply with the SysML standard, it is a logical extension that is quite useful when viewing requirements.
- With the addition of the new diagrams, the ER diagram, L0 physical block diagram, L0 interface block diagram, and FFBD have been toggled off explorer tabs and toolbars by default. All diagrams are still available from the pop-up views menu and can be added to the toolbars and explorer tabs as desired via the User Preferences.



Organizing, Navigating, and Focusing on Models

The introduction of the project explorer many versions back fundamentally changed the base navigation in CORE. Moving from a combination of browsers and editors to the explorer provided a familiar landing page and a single entry point for project navigation. With CORE 8, we have extended the explorer to introduce new navigation paradigms and increase visibility of information. The result is an explorer that remains familiar to current users but provides alternate – and we hope improved – paths into the system model.

Packages

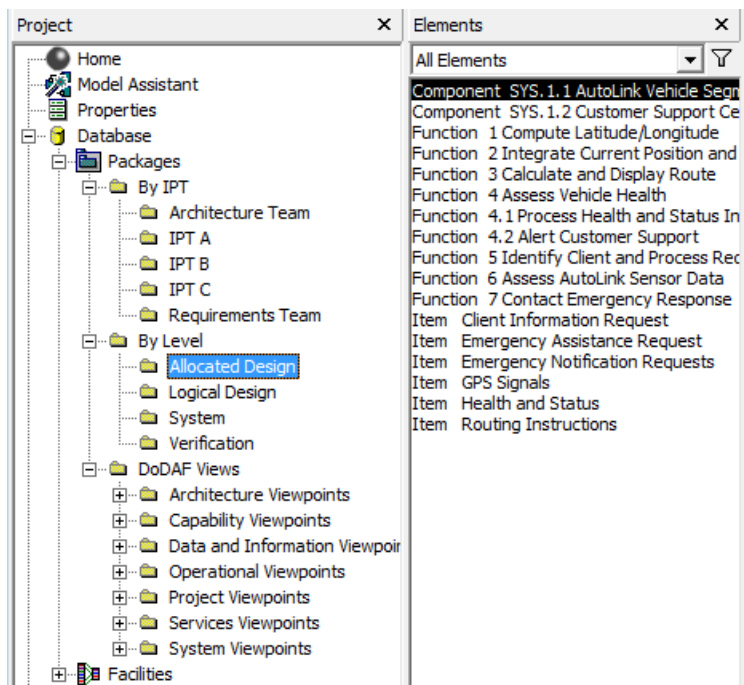
Traditionally, navigation within CORE has been database-centric. To access the element of interest, you started by drilling into the specific class and then down into folders until you found what you were looking for. For those with a database-centric frame of reference and command of the MBSE language, wonderful! Those without these pre-requisites could easily struggle.

With CORE 8, we raised the question of alternate navigation approaches, not to replace the database-centric paradigm but to complement it. We hypothesized role-based structures, level-based structures, and view-based structures (to name just a few). To enable all of these approaches, we have leveraged the SysML approach of packages.

In SysML, packages are fundamentally tools for model management. They are a type of container that can hold diagrams, elements, and other packages (creating a nested structure). CORE now includes a Package class that enables this behavior. As packages are created, they are shown in the project list pane, enabling you to implement whatever packaging and navigation approaches are desired.

Of particular note:

- Packages are shown as a segment under the Database item in the project list. This locates them alongside the existing ability to navigate the database via facility / class.
- Under the Packages item, CORE lists all root packages. These are the package elements that are not included in any other package.
- Packages and their contents can be manipulated directly from the Package section of the navigation list or by manipulating the native CORE elements in the Package class.
- Elements can be included in more than one package. This allows you to establish multiple package-based navigation schemes – by IPT, by level, ...
- As with element folders, packages can be set to show sub package contents. This allows you to look all the way down the package hierarchy in a single, integrated list. Elements included in sub packages are shown in italics.
- Package contents can span multiple classes. This enables you to apply operations across elements from multiple classes (such as setting access control, purging and baselining, etc.).
- Because package contents can span multiple classes, the table view is not available when navigating packages.



- A preference has been added to the general settings in User Preferences to allow you to specify where you want to “start” in the project explorer. Available options are the main packages selection as well as the dominant Vitech-defined facilities (architecture, essentials, systems engineering, program management, verification, and all classes). The default is the new essentials facility. This preference jumps you right into the middle of the model while still exposing you to the richness of the model.
 - Note: If the selected starting point is a facility, the Component class will be selected by default. This begins navigation at the system, subsystem, or component level, generally a logical point of departure.

Facilities

From the early days of CORE, we have made use of the concept of facilities to filter the complete collection of element classes down to the subset of immediate interest. Accessible from a drop-down in the explorer toolbar, selecting a different facility updated the list of classes – and therefore elements – that you could see.

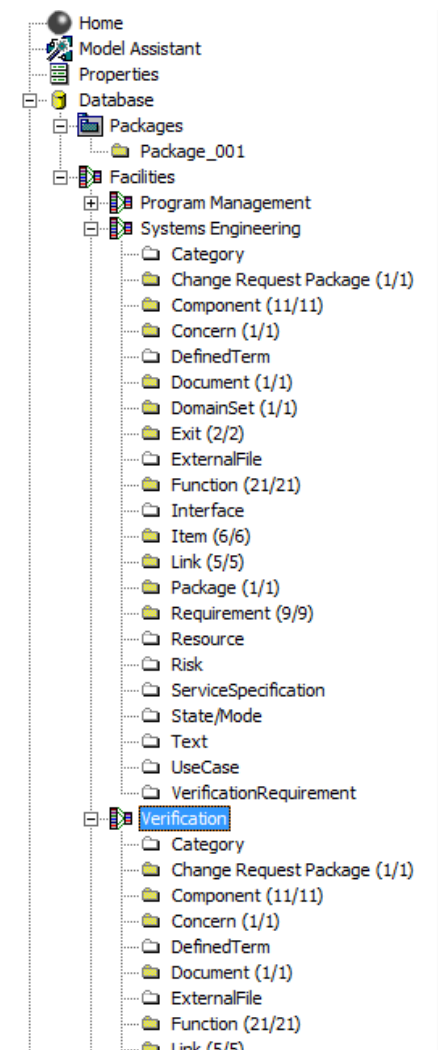
In CORE 8, the concept of facilities remains but the navigation approach has changed. Rather than selecting the facility of interest from a drop-down menu in the toolbar, all available facilities are now shown as part of the navigation tree in the project list pane. The intent is to seamlessly integrate this concept as users navigate their model while also increasing visibility of other available facilities (most notably verification and program management).

By default, a project opens with the facility list shown. You can then expand whatever facility – or facilities – are of interest for the analysis or communication at hand. Everything else with respect to facilities and database-centric navigation has remained the same.

Additional Explorer Content

In addition to the introduction of packages and the expanded visibility of facilities, three other additions have been made to the project navigation tree:

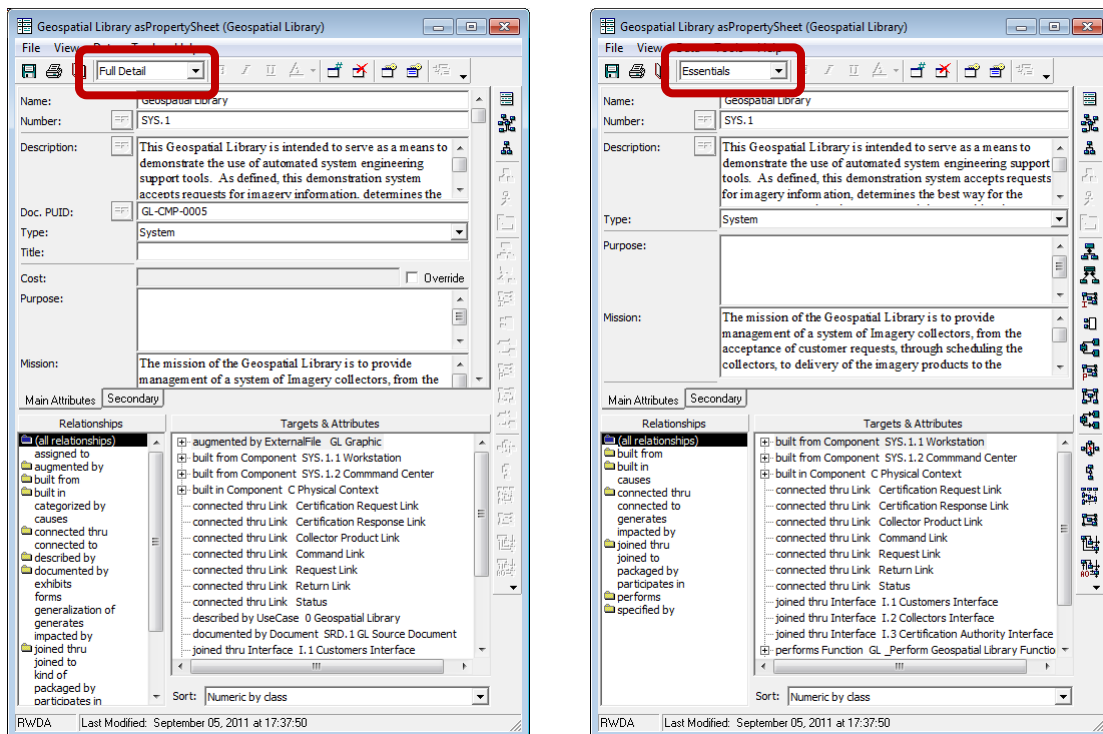
- **Model Assistant.** The model assistant options (discussed later in this feature guide) are accessible via the explorer. This both increases visibility and provides additional screen space to describe the capabilities.
- **Perspectives.** Perspectives (discussed on the following page) are a new dimension to the schema intended to lower the barrier to entry for MBSE.
- **Icon Templates.** Long buried in the User Preferences, icon templates were previously a per-user setting. With the changes in CORE 8, icon templates are now a per-project setting. By moving the icon templates to the Utilities section of the explorer, we hope to increase the accessibility of this flexible representation tool.



Perspectives

One of the challenges of MBSE is simply the language itself. To have the richness necessary to represent the various aspects of the system model requires a rich interconnected language of classes, relationships, and attributes (not to mention the underlying structures). Navigating this language was once described as using a jeweler's loupe to inspect your system, giving you great insight at the detail level but making it very difficult to abstract and understand at the higher level.

Perspectives allow you to focus on the aspects of the system model – specifically of an element – that are most relevant to you at the current time. Essentially, they allow you to specify what attributes and relationships you want to see in an element's property sheet. Perspectives can be role-based (requirements analyst, architect, test manager, program manager), level-based (system context, intermediate, full-detail), or anything-you-want-based. You can quickly move between perspectives to get the right view of the model right now, focusing and progressively elaborating as desired.

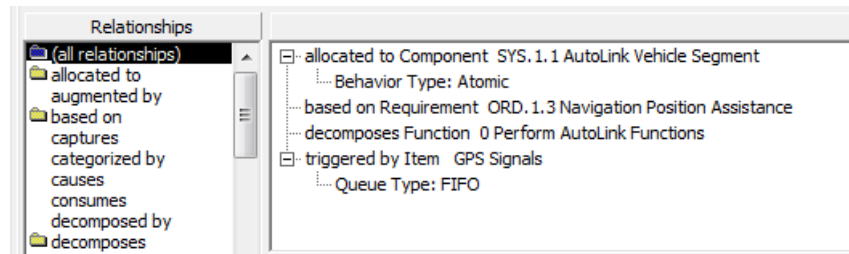


Perspectives operate on the explorer property sheet, stand-alone property sheets, and the element extractor. New perspectives can be created and non-essential perspectives redefined while in schema mode. Five perspectives are currently pre-defined:

- **Full Details.** The Full Detail perspective displays all attributes and relationships you a comprehensive view of the system model. This perspective maps to the traditional CORE representation.
- **Essentials.** The Essentials perspective displays the foundational attributes and relationships. This perspective is a point of departure to the full richness of the MBSE language.
- **System Context.** The System Context perspective offers the high-level view placing the design in context. It is intended to offer a view of the system in its setting so that it can be seen in "context."
- **Progress Check.** The Progress Check perspective is designed to answer the question "Where am I?" with respect to the design. It shows how complete the design is in terms of domains and detail.
- **Project Manager.** The Project Manager perspective is for the Project Manager who is not concerned about the nitty gritty detail of the engineering design as much as design aspects necessary to manage the progress of the project.

Additional Navigation Changes

- The element property sheet has been extended to include an “all relationships” option in the relationships list pane. This provides a compact textual representation of how the element is connected to the rest of the system model.



- The default for “expand relationship attributes” has been changed so that relationship attributes are no longer expanded by default in attribute lists. Relationship attributes are a powerful part of the model, but they are certainly the icing on the cake. As more relationship attributes have been added to support SysML and enrich the model, they block insight into the connectivity of the model. This is particularly true where the new “(all relationships)” selection is used. Users who wish to see these attributes by default can easily toggle this setting back on. All users still see the tree representation with the plus sign indicating the attributes are present.

Assisting the Analyst

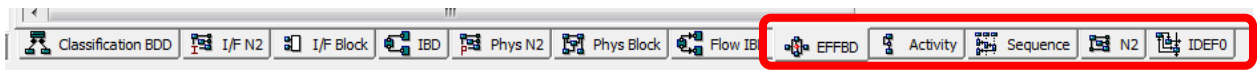
“Model Assistant” represents a fundamentally new capability in CORE as we begin to branch into truly assisted engineering. The intent is to better highlight the power and integration of a true model-based approach while reducing some of the framework details required to successfully implement MBSE. The initial capability provides a powerful foundation that we expect to build upon going forward.

The individual model assistant rules are enabled or disabled per-user via the Model Assistant item in the project explorer tree. By default, all options are enabled.

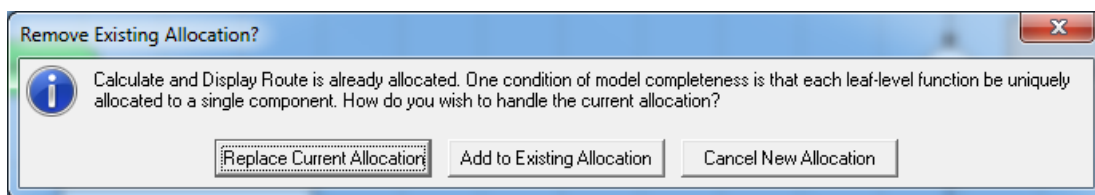
- Show Root Function Views.** Over the last nineteen years, we have focused attention on an underlying integrated model represented through multiple specialty representations. While the connection between diagrams within a domain (behavioral or physical) has been clear, the connection between the functional and physical aspects has not. Ideally, much as one can pick up a cube and twist it to look at any face, we would be able to “pick up” the system model and rotate it to see any perspective.

Reducing the learning curve while reinforcing the connected nature of the underlying model, users can now open logical views on physical objects. If the physical object has a root function, the logical view transparently opens on that root function. The net result is simplicity of operation (individuals often think of a system in terms of its components) with clear indication that logical and physical representations are two perspectives of a single system model.

To see this capability in action, import the AutoLink example and click through the Component class. For components with root functions, you can see the many logical views. For components without a root function, you will receive an appropriate message when attempting to access a logical view.



- Prompt on Multiple Allocation.** One condition of completeness is that each leaf-level function be uniquely allocated to a single component. With this option enabled, CORE will prompt when you attempt to allocate a function to multiple components. You will have the option of completing the multiple allocations, replacing the existing allocation with the new component, or canceling the action. This streamlines the allocation process while helping to maintain model completeness.



- Auto-create Root Functions.** Root functions are an essential connector between logical and physical aspects of the model. To those who understand the methodology, their role is clear. To those learning, they are odd artifacts of the approach that muddy the system definition effort.

CORE 8 (optionally) auto-creates a root function when you create a component. The function is named “_Perform ABC Functions” where ABC is the name of the component. This root function houses behavior decomposition for the component. After creation, this root function can be manipulated as desired – renamed, related, or even deleted. However, the up-front automation reduces the learning curve, largely moving this “perspiration” step behind the scenes.

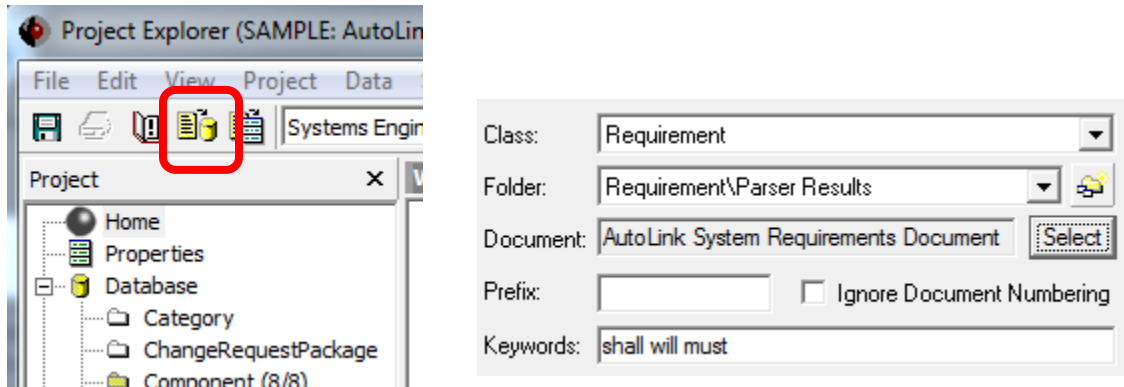
- **Auto-allocate on Decomposition.** As models are progressively expanded to a greater level of detail, flow-down of conditions from the parent level of the model are important. With this option enabled, CORE automatically allocates child functions when they are inserted in the decomposition. You can change this allocation, but it provides an initial flow-down from the parent allocation to serve as a guide.
- **Auto-relate on Allocation.** After the integrated behavior is completed at a layer, the behavior is allocated to components where component logic is developed. Previously, the allocation and behavior were disjoint steps, subject to cross-check via consistency scripts but without automated support.

CORE 8 (optionally) auto-inserts a function into the root function decomposition when you allocate that function to the component. This only occurs if neither the function nor one of its ancestors is already on the root function decomposition. Again, this automation helps reinforce the alignment, integration, and synchronization of the models. It does not create the behavior structure, but it does ensure all necessary functions are present and considered.

Accelerating the Initial Requirements Phase

For years Vitech has emphasized the need for a human in the loop during the requirements parsing process. Relying purely on automated parsers makes it too easy to lose critical context and miss implied requirements, placing the entire project at risk. CORE 8 includes an automated parser designed to take much of the manual labor out of parsing so that the analyst can focus on the content.

The document parser is opened by clicking the parser icon on the main toolbar. Once opened, you must load a document (.DOC, .HTML, or .TXT) much as you would with the manual element extractor. The definition pane below the document allows you to set multiple parameters:



- Class – the class of the elements to be created. Requirement is the default class, but any class can be selected.
- Folder – the folder in which to place the parsed elements. A best practice is to place requirements in a separate folder for thorough review to avoid the garbage-in, garbage-out problem. Note that a new folder can be created directly from the definition pane by clicking the folder button.
- Document – the Document element that will be linked to the root elements parsed from the document. If no document is specified, parsing will continue and root elements will simply not trace to a document.
- Prefix – a string that will prefix the names of all elements created during the parsing process.
- Ignore Document Numbering – where the source document aligns well with hierarchical numbering rules, CORE will interpret paragraph numbers to build a complete hierarchy that corresponds to the document hierarchy. Where this is not the case, enabling the Ignore Document Numbering option will auto-parse the document without generating any implied hierarchical structure. Parent-child relationships will be generated between sentences and their enclosing paragraph, but no other parent-child relationships will be created. You can then easily establish the desired hierarchical requirements structure after the elements have been created in the database. (Note: When using this option, it is recommended that no document be specified in the Document field. This avoids any rework to break relationships between requirements and the source document as a requirements hierarchy is created.)
- Keywords – the collection of words the parser will key on to separate requirements from other content. The generally accepted set is shall, will, and must. Note that if no keywords are specified, you will be prompted for confirmation before the document is parsed. If you choose to proceed, all content will be identified as elements rather than “debris”.

Once you specify the required settings, parsing the document creates temporary results displayed in a table view. This table view is fully editable, allowing you to review – and change – the definitions before the content is entered into the database.

	Requirement	Number	Name	Description	Paragraph Number	Paragraph Title	Type	refines
1	<input type="checkbox"/>	1.0	SCOPE		1.0	SCOPE	nil	
2	<input type="checkbox"/>	2.0	APPLICABLE DOCUMENTS		2.0	APPLICABLE DOCUMENTS	nil	
3	<input checked="" type="checkbox"/>	3.0	GENERAL REQUIREMENTS		3.0	GENERAL REQUIREMENTS	nil	
4	<input checked="" type="checkbox"/>	3.1	Specific Requirements		3.1	Specific Requirements	nil Requirement 3.0 GENERAL REQUI
5	<input checked="" type="checkbox"/>	3.1.1	Specific Requirements 1	The AutoLink System shall automatically contact the Customer Support Center within 1 minute in	3.1	Specific Requirements	nil Requirement 3.1 Specific Require
6	<input checked="" type="checkbox"/>	3.1.2	Specific Requirements 2	The AutoLink System shall contact emergency personnel within 2 minutes of the diagnosis of an	3.1	Specific Requirements	nil Requirement 3.1 Specific Require
7	<input checked="" type="checkbox"/>	3.1.3	Specific Requirements 3	The AutoLink System shall receive GPS transmissions for navigation.	3.1	Specific Requirements	nil Requirement 3.1 Specific Require
8	<input checked="" type="checkbox"/>	3.1.4	Specific Requirements 4	The AutoLink System shall wirelessly communicate between the AutoLink's remote components	3.1	Specific Requirements	nil Requirement 3.1 Specific Require
9	<input checked="" type="checkbox"/>	3.1.5	Specific Requirements 5	The AutoLink System shall have a Mean-Time-Before-Failure of 20,000 hours and a	3.1	Specific Requirements	nil Requirement 3.1 Specific Require
10	<input checked="" type="checkbox"/>	3.1.6	Specific Requirements 6	The AutoLink System shall have an Availability of 0.9995.	3.1	Specific Requirements	nil Requirement 3.1 Specific Require
11	<input type="checkbox"/>	a	Debris 16	a. Vehicle operator needs directions to a location,	3.0	GENERAL REQUIREMENTS	nil Requirement 3.0 GENERAL REQUI
12	<input type="checkbox"/>	a.1	Debris 17	a. Vehicle operator needs directions to a location,	3.0	GENERAL REQUIREMENTS	nil Requirement a Debris 16
13	<input type="checkbox"/>	b	Debris 18	b. Vehicle operator, police and rescue services needs to locate vehicle	3.0	GENERAL REQUIREMENTS	nil Requirement 3.0 GENERAL REQUI
14	<input type="checkbox"/>	b.1	Debris 19	b. Vehicle operator, police and rescue services needs to locate	3.0	GENERAL REQUIREMENTS	nil Requirement b Debris 18

The number, name, description, paragraph number, paragraph title, and refines relationships are each created per the document parsing rules. The individual attributes – along with the requirement type – can be edited directly in the table before the elements are created.

- The initial column indicates whether or not this object is a requirement. Objects are flagged as requirements if they include one of the specified keywords. The requirement status for individual objects can be changed.
- Individual rows or collections of rows can be deleted simply by selecting the row header and right-clicking to select the Delete Elements command.
- The table can be sorted in the same way as other CORE tables (either by double-clicking a column header or by using the Sort by Columns command from the menu).
- Likewise, though the refines relationship is displayed, it cannot be changed. Because these objects have not yet been created, it is not possible to relate them to one another. Therefore, while the table view is extremely valuable for reviewing and adjusting parsing results in the context of the source document, heavy-duty analytical work, requirements allocation, and restructuring should be done after the elements are created in CORE.

After the document has been auto-parsed and the results manipulated as desired, you have two options available for entering the results into the database:

- **Create All Elements.** This command creates CORE elements for every parsed row, regardless of whether or not the row is marked as a requirement.
- **Create Requirements.** This command creates CORE elements for all rows marked as requirements. The other rows are treated as non-requirement content and discarded.

Tip #1: Better to create debris now and clean it up than to regret the loss later. Using the proper set of keywords will help flag (and name) content appropriately. Upon further review, you can assess whether “debris” truly is information that can be discarded or if it contains a poorly stated or implied requirement.

Tip #2: Use the interim results for a preliminary review of parsing success then create the elements and use the project explorer (either with the table view or property sheets) to do heavy editing.

A Myriad of Additional Refinements

In addition to the primary features added and user experience enhancements made in CORE 8, many other enhancements have been provided. Many of these are in response to wonderful suggestions from our user community.

Scheduling Tasks

As much as we want to focus our days around creativity and innovation, often times “housekeeping” tasks can consume much of our time and our mental energy. A daily web update to publish interim project deliverables, a key metrics report to log our status, regular data exchange / synchronization with the greater engineering project environment, or simply utilities to aid the engineer – all offer value but at the expense of the project at hand.

With CORE 8, you now have the ability to schedule tasks (specifically reports), relying on automation to address the perspiration of the project while you focus your precious time on the value-added inspiration. Part of the user preferences, the task scheduler allows you to identify specific reports to run at any frequency for whatever need you have.

The image shows two overlapping windows from the CORE 8 software. The background window is the 'User Preferences' dialog, with the 'Task Scheduler' option selected in the left-hand tree view. The main area of this window displays a 'Scheduled Tasks' table with the following data:

Name	Schedule	Description
Consistency Checks	Every 01:00	Model completeness /
Daily HTML Output	MTWThF at 23:00	Project web publication
PME Exchange	At login	PME exchange file

Below the table are buttons for 'New Task', 'Edit Task', and 'Delete Task'. The foreground window is the 'Edit Scheduled Report Execution' dialog box. It contains the following fields and options:

- Name: Daily HTML Output
- Description: Project web publication
- Project: Geospatial Library
- Report: (All Reports)
- Script: HTML Report v1.17
- Schedule:
 - At login
 - At 23:00
 - M T W Th F S Su
 - Every [] while logged in

Clicking the new task button opens a job scheduling dialog. For each scheduled task, you provide:

- Name – a short name to help in managing your automated tasks.
- Description – a longer descriptive string describing the scheduled task.
- Project – the root project for the report. You may choose any project from the drop down list of all projects you have access to in the local and remote repository. Every scheduled task must have a root project specified. Using the advanced capabilities in scripting, you may access other projects as well.
- Report – the folder and script drop down lists allow you to navigate to the specific script you want to execute. This can range anywhere from a data exchange script you run every 15 minutes to a set of consistency checks run hourly to a formal deliverable report produced every Friday night.
- Schedule – the definition of when the report should run. Options include at login, at a fixed day and time, or at a set frequency.

Any CORE report can be scheduled to run as a scheduled task. Prompts will use their default values and output files will be auto-saved in the CORE\Output\temp folder so that existing reports can run unattended. If you want special behavior – writing a file to a network drive for data exchange or publishing a work product to a specific location – you can easily modify a script to do just that. The intent is to seamlessly automate existing capabilities while at the same time enriching your toolkit to enhance and customize the CORE experience.

All scheduled reports run within your user session while you are logged into CORE. This means that they run with your privileges and are governed by all project permissions.

Tip #1: When scheduling a report, be sure to run the report yourself first. As noted in the Polishing Reports section below, CORE reporting prompts now remember your inputs and use these as defaults for the next execution. Scheduled reports take advantage of this capability, leveraging your last inputs when running unattended.

Tip #2: As with graphics on diagrams, resist the temptation to overdo a good thing. Though these tasks run in the background, they will compete with your other programs running on your machine for processor time. If you have a reasonable machine, you won't even notice reports running in the background. If your machine is overloaded, schedule reports to run over lunch, during a regular staff meeting, etc.

Tip #3: If you make regular use of automated reports, place a shortcut to your CORE\Output\temp directory in your Windows Document folder or other favored location for quick access to your outputs.

Polishing Reports

Generating outputs directly from the system model is a key aspect and critical benefit of using CORE. With this release, we are not seeking radical changes in our reporting mechanism. Instead, the objective is to enhance the power, usability, and end-user experience by building upon the current foundation.

- **Remembering user inputs.** Users frequently reuse the same capabilities day in and day out. In particular, users frequently run and re-run the same reports as part of a daily/weekly routine or in the final push to deliver for a major milestone. In doing so, re-entering the same data in the various reporting prompts becomes tedious. Worse yet, if the objective is to re-run the report in the same configuration, you have to remember how you answered every question in order to generate comparable results.

CORE 8 now transparently remembers how you answer a prompt and re-uses that value the next time the report is run. This is per-user, per-prompt, and per-report meaning that individual user responses are remembered and different responses to different reports are remembered.

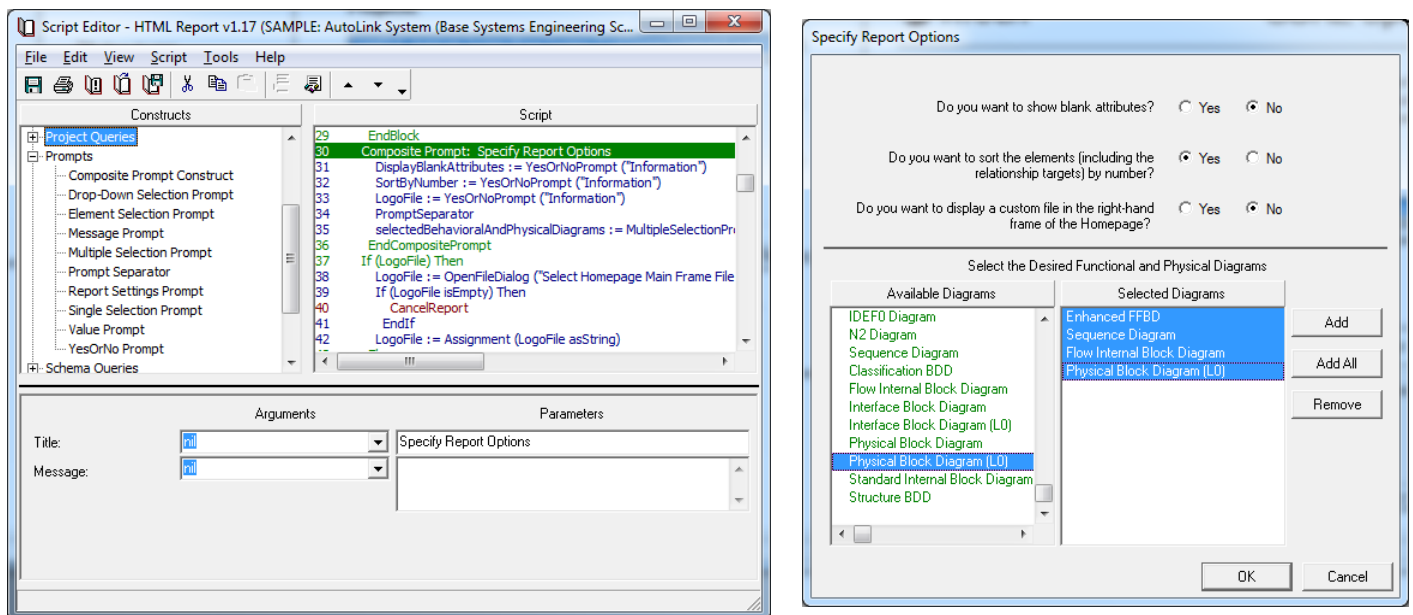
Once a report is “converted” to CORE 8 format, nothing special is required – either by the script author or the end-user – to make this happen. All “converting” means in this case is simply opening and saving the report in CORE 8. In doing so, CORE automatically inserts the metadata required to track user inputs.

CORE 8 New Features Guide

Note: The data pertaining to user selections is stored as part of your user profile and is available either as part of their local settings or their server account. However, it is not exported with the project or user settings. Also, if a prompt specifies a default value, that default value takes precedence over the prior user response when CORE next runs the report. The rationale is that if a script author specifies a default, there is a reason for doing so, and this reason should not be lost.

- **Composite prompts.** While CORE makes a rich set of prompts available to the report author – single values, drop downs, lists, element lists, etc. – each prompt is separate and self-contained. The prompts are designed to ask one question and get one answer. As a result, report authors are forced to rely upon multiple prompts to get the input they need and offer the end-user the flexible configuration they want. Unfortunately, this often results in death by prompts when running a report as users face a seemingly endless chain of dialogs.

CORE 8 introduces a composite prompt block that allows report authors to group their existing prompts into one or more combined prompts. When the script is run, CORE assembles these individual prompts into an ordered series more natural for the end-user, much like filling out any online form. No customization of existing prompts is required. Simply insert the new composite



prompt block, drag in your existing prompts, and you are ready to roll.

Additional Report Changes

- A construct has been added to open a temporary file. This allows CORE to auto-name the file and place it in a special TEMP folder in the output directory. For files that are intended for interim use, this construct allows scripts to avoid prompting you to specify the output file. In particular, many scripts that present on-screen results (consistency checks, queries, etc.) can be rewritten to present the output as formatted RTF results opened in Word rather than writing to the ASCII script transcript.
- A construct has been added to access random numbers. Direct access to the random number distributions enables many advanced simulation scenarios.
- The DiagramOutput and DiagramFileOutput constructs have been extended to specify the display mode (relating to hiding and showing content). The default is to use the stored view which mirrors the last displayed state in CORE. However, providing explicit access to the various diagram

modes allows custom reports tailored to specific needs (customer communication, internal review, etc.) without concern for the current representation state in CORE.

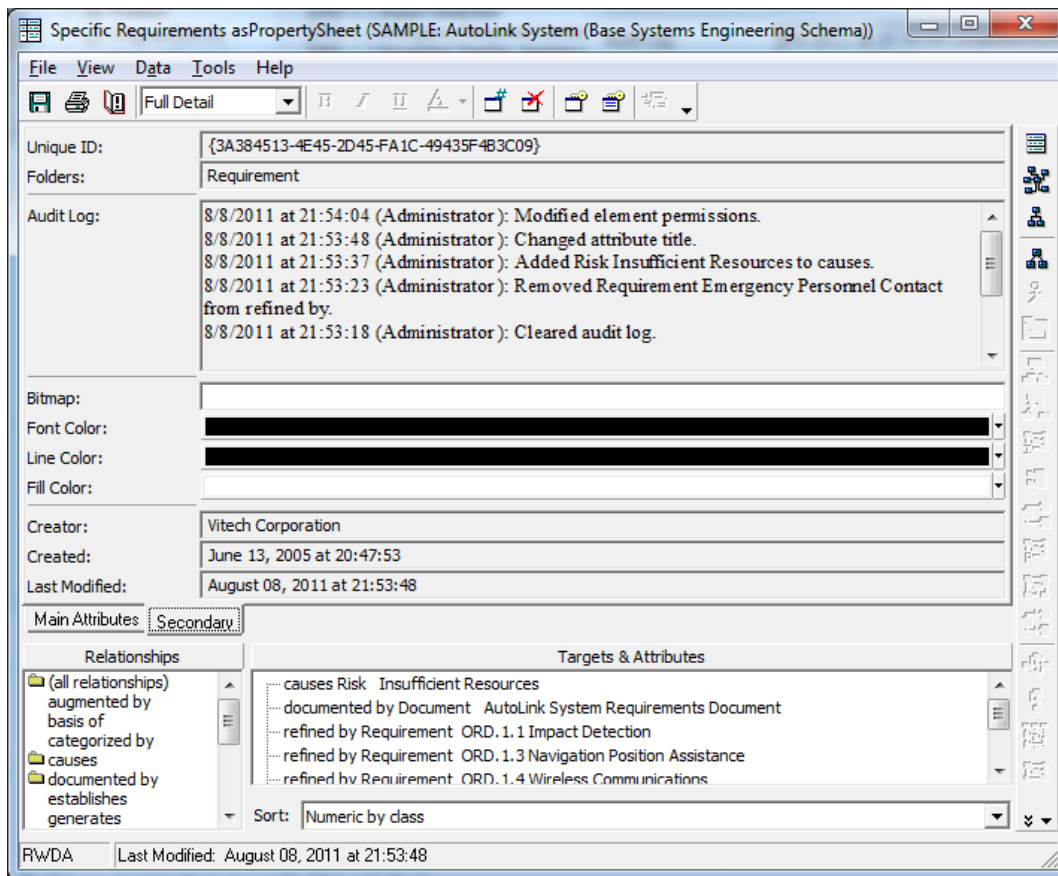
- Legacy script support for CORE 4 schemas has now been removed. The CORE 4 schema lacks many classes and relationships relevant to current capabilities and representations, and CORE 5 schemas date back to 2005. Removing support for this outdated schema encourages any remaining projects to migrate while eliminating distracting outdated options.
- Failure to close script input or output files no longer logs a conflict at the end of script execution.

Audit Logs

CORE 6 introduced versioning at the attribute level. Projects can define which attributes are to be versioned and when versioning is enabled. This provides the finest level of control, but it does not provide greater history at the element level.

CORE 8 introduces audit logs. Much as with versioning, audit logging is enabled or disabled at the project level. Once enabled, audit logging maintains a basic textual log of all changes made along with the timestamp and the responsible user. These are stored in a read-only auditLog attribute on the secondary tab of the property sheet. Combined with the highly detailed attribute versioning and the project comparison reports, audit logs provide insight into the evolution of an element.

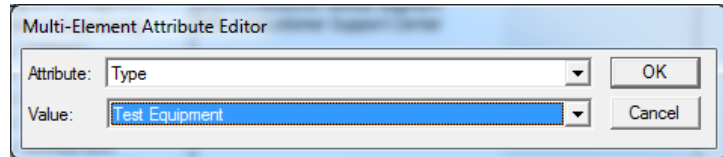
Much as with versioning, commands exist at the element, category, package, and project level to clear audit logs. Clearing the audit log requires element administrator privileges and enters a corresponding entry in the audit log history. Audit logging is not appropriate at the very start of a project, but as the project matures and controlled evolution becomes appropriate, audit logging should be enabled.



Setting Attributes for Multiple Objects

Often times, there is a need to set a specific value for a range of elements – most commonly, setting the type for requirements, components, etc. While the grid view provides a wonderful way to see the state and quickly navigate through the elements to make the change, changing 10 elements still requires entering the same change 10 times.

The Set Attribute command allows you to change an attribute value for one or one hundred elements simultaneously. It can be applied to a homogenous set of elements in a single class or a heterogeneous set of elements in a package or on a diagram.

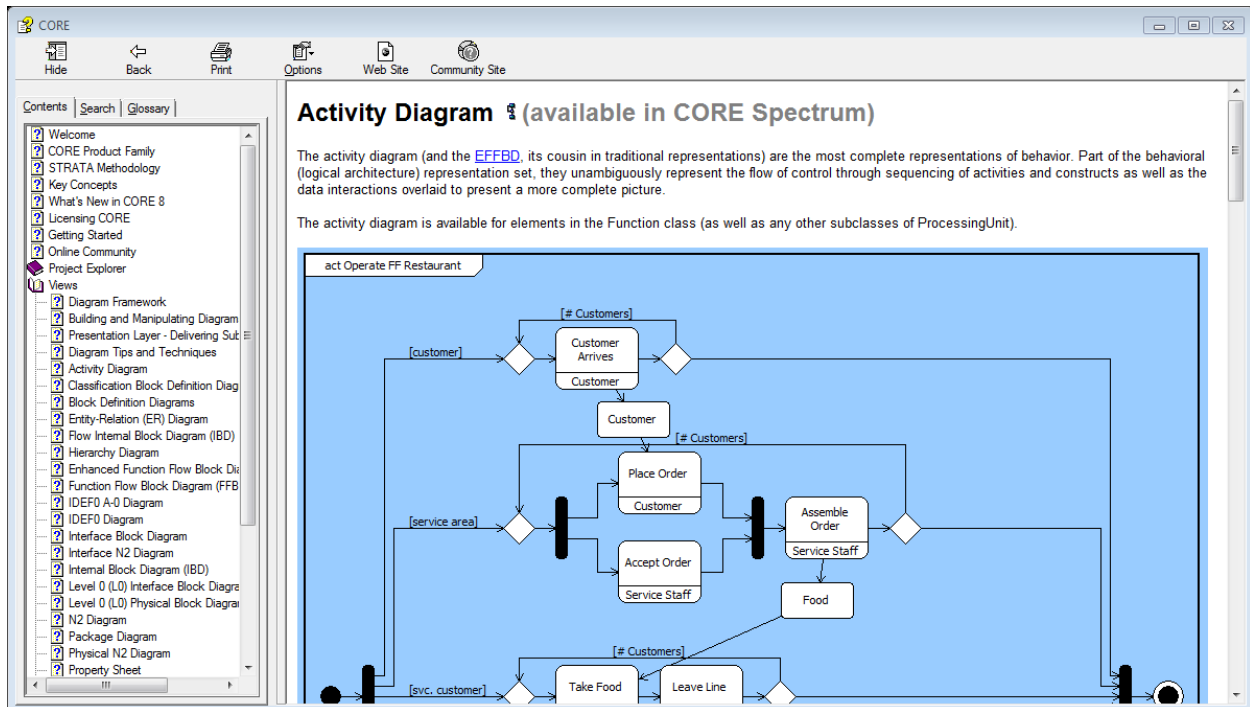


Simply select the elements in question (via shift or control clicking as appropriate) and click the Set Attribute command. The resulting dialog keys off the first element selected to present the list of possible attributes to set. Select the desired attribute from the drop down and the value pane will change to best represent the attribute type and to show the current value. When you click ok, that value will be set for all selected elements.

Tip: The Set Attribute command has been added to menus and pop-up menus across CORE. This includes element lists and diagrams. Often times, even when addressing a single element, it may be quicker to use this command than to open the property sheet to make the desired change.

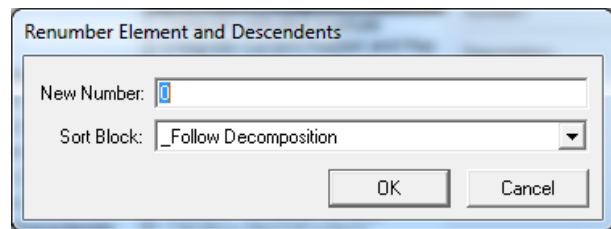
Updated Online Help

To accompany all of the other changes and enhancements in CORE, we have completely rewritten the online help from the ground-up. While reading a help file won't make you a systems engineer, the online help is an excellent guide to better understanding CORE's concepts and commands.

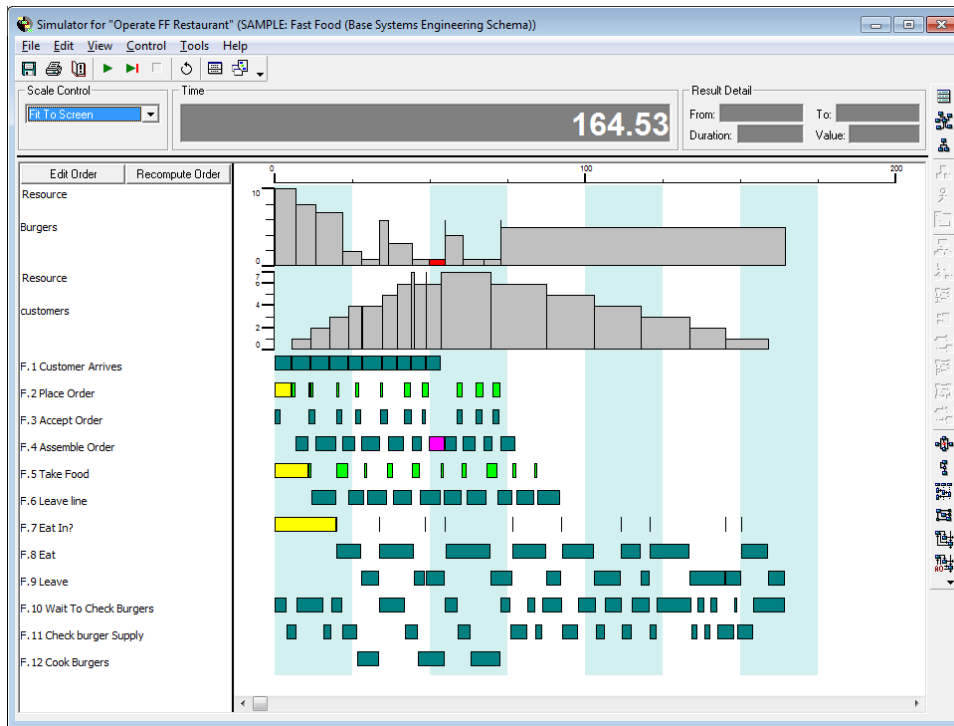


Additional Changes

- Projects
 - System administrators now have the ability to delete any project in their repository regardless of project-level permissions. In the event that the system administrator does not have read permission, the project will be displayed as UNKNOWN in the Administrative Tools. This provides system administrators the full control necessary to manage the repository without violating access control. (Note that this is the only list that shows projects for what the system administrator does not have read access.)
- Import / Export
 - Imports of change files now disregard the following project-level parameters: creator, creationStamp, maintainMergeHistory, useVersioning, accessControlList. These project parameters are aligned either with project creation or with administrative action. Disallowing their import from change files reduces the possibility of errors during the integration effort.
 - The access control list from an import file is now applied as the final step of the import process in the case of a repository or project export.
 - The users and groups have been removed from the project export since users are part of the repository, not a specific project. (Note that while this avoids the potential proliferation of users and groups, this now creates an external dependency for project exports that must be properly managed.)
 - Import Users & Groups and Export Users & Groups commands have been removed from the Administrative tools menu. These commands have been superseded by the XML import / export capability and therefore removed to avoid user confusion.
- General GUI
 - When renumbering an element with a structure, you now have the option of using the decomposition structure (default) or a sort block.
- User Preferences
 - The Preferences Dialog is now resizable.
 - The default file locations for data, source, and output files have been removed from the User Preferences (though the file structure remains for those users who choose to use it). These options were largely outdated, particularly since the default folders are in the Program Data location rather than the Windows user folder. Most users organize their files their way, so operationally this preference (which took users back to specific file folders rather than their last location) was more of a hindrance than a help.
 - Note: Navigating to external references and external images will still use the base paths specified at the project level. This helps the individual and the team keep their key files organized rather than distributed.
 - Enablement / disablement of RTF support in text attributes can now be handled via the general settings in the User Preferences. This simplifies configuration of this option (primarily required by our Korean users). Note that if this setting is changed, you must close and restart CORE before the new setting takes effect.
- Hierarchy Definitions
 - A Package hierarchy definition has been added to the default set. For those making use of packages – either to organize or to navigate their data – the package hierarchy provides a multi-level visualization of package contents.



- COREsim
 - The timeline view now has a banded background. Alternating colors are drawn in vertical bands based upon the minor tick mark width. These color bands help align the eye vertically between time increments and various events on the timeline. (Note that this banding is only present within the CORE window for analysis purposes and is not shown when saving the timeline to a graphic file or when printing the timeline.)



- Double-clicking any block on the timeline now opens a property sheet on the corresponding element. This simplifies navigating to the element of interest to review or revise attributes and relationships of interest.

Expanding the Model

With each release, Vitech continues to enhance the underlying schemas to reflect our lessons learned and best practices in the application of model-based systems engineering. The v80 versions of our base and DoDAF schemas include revisions and extensions in the areas of:

- Change management – to better support and integrate with your project and corporate change review / change control processes
- Test and evaluation – to deliver new capabilities in support of this critical aspect of the project lifecycle
- Program management – to help close the perceived and often real gap between program management and systems engineering

A complete list of all schema changes is included at the end of these release notes. At the highest level, the key schema changes for all schemas (including the legacy v60 and v70 schemas) include:

- A new Package class (supported by a new packages / packaged by relation pair) to parallel the existing Category class for organization and navigation purposes
- A new auditLog attribute on Element (and inherited by all classes) to support an automatically maintained, human-readable textual representation of changes made to an element
- A new bitmap attribute on Element (and inherited by all classes) to support graphical images on diagrams as an alternative to geometric frames.

The key schema changes of note for the v80 base systems engineering and DoDAF 2.0 schema include:

- A new ChangeRequestPackage class (supported by a new originated by / originates relation pair) to better support tracking change requests and their impacts in CORE
- New TestItem and TestActivity classes to model required T&E activities
- Addition of a new impacted by / impacts relation pair to better relate Issues, Risks, and ChangeRequestPackages with the affected portions of the model
- Aliasing Issue to Concern to improve alignment between program management and systems engineering language
- Revision of the Risk likelihood attribute from a numerical value to a low / medium / high scale

Note: With the release of CORE 8, support for all CORE v40 and CORE v50 schemas has been sunset. If your project is still using one of these schemas and would like an assessment of the impact of migrating to a v60, v70, or v80 schema, please contact Vitech Customer Support (support@vitechcorp.com).

Migrating Projects from pre-v80 Schemas

Given the nature of the schema changes made in the v80 base and DoDAF schemas, some project migration is required to transform specific classes and specific attributes. To best support this migration, CORE includes specific schema extension files and reports which combine to form an extensible migration toolkit. This toolkit is targeted at the standard Vitech schemas. If your project uses a customized schema, additional migration support may be required. If you have made schema extensions and would like to understand what special steps – if any – are required to support your extensions, please contact [Vitech Customer Support](#) with a copy of your extensions. We are happy to review these and advise you in your migration accordingly.

Note: You do **not** have to migrate your v60 or v70 schemas in order to make use of CORE 8. You should assess your project needs, your project lifecycle, and the changes present in the v80 schemas to determine if, and when, to migrate your schemas. Projects nearing a major milestone or approaching conclusion should strongly consider remaining with their current schema. We recommend that others consider moving to the v80 schemas to take advantage of the latest improvements in the MBSE language.

Comprehensive List of CORE 8 Schema Changes

General Changes Included in All Schemas (including Legacy Schemas)

- New classes
 - Package (parallel class to Category for navigation purposes)
- New relation pairs
 - packages / packaged by
- Class changes
 - Element
 - Added auditLog to support an automatically maintained, human-readable textual representation of changes made to an element.
 - Added bitmap attribute to support bitmaps on icons

Systems Engineering Schema Changes (also reflected in the DoDAF 2.0 Schema)

- New classes
 - Nexus (abstract superclass for Issue and Concern)
 - ChangeRequestPackage
 - Package (parallel class to Category for navigation purposes)
 - ServiceSpecification
 - TestItem
 - TestActivity
- New relation pairs
 - established by / establishes
 - executed by / executes
 - impacted by / impacts
 - originated by / originates
 - packages / packaged by
- relationship changes
 - Revised the extends relationship to support unlimited targets.
 - Added a type attribute to the kind of relationship
- Class changes
 - Element
 - Added auditLog to support an automatically maintained, human-readable textual representation of changes made to an element.
 - Added bitmap attribute to support bitmaps on icons
- Component
 - Added clin attribute
 - Added possible values to the type attribute
 - Family of Systems
 - Network
 - Service
 - Software Item
 - SW Element
 - SWCI
 - System Architecture
 - System of Systems
 - Added impacted by relationship with target classes Issue and Risk
- ConnectingUnit
 - Added impacted by relationship with target classes Issue and Risk
- DecomposableElement
 - Added impacted by relationship with target classes Issue and Risk

CORE 8 New Features Guide

- Document
 - Added clin attribute
 - Added possible values to the type attribute
 - Agreement
 - Drawing
 - Goal
 - Guidance
 - Information Asset
 - Procurement Specification
 - Risk Mitigation Plan
 - Service Level Specification
 - Statement of Work
 - Strategy
 - Threat
 - Added impacted by relationship with target class classes Issue and Risk
- DocumentFormat
 - Removed cageCode attribute
 - Removed contractNumber attribute
- Interface
 - Added impacted by relationship with target classes Issue and Risk
- Issue
 - Aliased to Concern
 - Changed fill color to tan
 - Renamed severity attribute to importance (implemented on new Point class)
 - Added target class UseCase to generated by relationship (implemented on new Point class)
 - Added impacts relationship with the following target classes (implemented on new Point class)
 - ConnectingUnit
 - DecomposableElement
 - ImplementationUnit
 - Interface
 - Organization
 - ProgramElement
 - Requirement
 - Resource
 - State/Mode
 - UseCase
 - VerificationElement
- Organization
 - Added generates relationship with target class ChangeRequestPackage
 - Added impacted by relationship with target classes Issue and Risk
 - Added originates relationship with target class ChangeRequestPackage
- ProgramElement
 - Added impacted by relationship with target classes Issue and Risk
 - Added accomplished by relationship with target class TestActivity
- Requirement
 - Added possible values to the type attribute
 - Incentive Award Fee Criterion
 - Programmatic
 - Test
 - Added establishes relationship with target class TestActivity
 - Added impacted by relationship with target classes Issue and Risk
- Resource
 - Added impacted by relationship with target classes Issue and Risk

- Risk
 - Changed fill color to red
 - Changed likelihood attribute type from float to enumeration
 - Low
 - Medium
 - High
 - Added possible values to the type attribute
 - Safety
 - Security
 - Added riskEffectiveDate attribute
 - Added statusDescription attribute
 - Added impacts relationship with the following target classes
 - ConnectingUnit
 - DecomposableElement
 - Document
 - ImplementationUnit
 - Organization
 - ProgramElement
 - Requirement
 - Resource
 - ServiceSpecification
 - State/Mode
 - VerificationElement
- State/Mode
 - Added impacted by relationship with target classes Issue and Risk
- UseCase
 - Added impacted by relationship with target classes Issue and Risk
- VerificationElement
 - Added impacted by relationship with target class Issue
- VerificationRequirement
 - Added executed by relationship with target class TestActivity
- Facility changes
 - Document Management
 - Added ServiceSpecification
 - Essentials Facility (new)
 - Component
 - Document
 - Function
 - Requirement
 - Risk
 - UseCase
 - VerificationRequirement
 - Systems Engineering
 - Added ServiceSpecification
 - Verification Facility
 - Added DefinedTerm
 - Added Product
 - Added ProgramActivity
 - Added ProgramElement
 - Added Resource
 - Added Risk
 - Added ServiceSpecification
 - Added State/Mode
 - Added TestActivity
 - Added TestItem

THIS PAGE INTENTIONALLY BLANK



Vitech Corporation
2270 Kraft Drive, Suite 1600
Blacksburg, Virginia 24060
540.951.3322 FAX: 540.951.8222
Customer Support: support@vitechcorp.com
www.vitechcorp.com