Framework & Model Management System
For Enterprise Architect

Introduction

Anyone trying to develop a modeling environment within their organization has been faced with many competing demands. The demands come from those responsible for establishing architectural guidance for the organization, the industry standard frameworks that are designed to assist in establishing the architectural direction, the features provided by the available toolsets in this space, and those who are impacted by the architectural direction as they attempt to do their daily work.

There are heated debates about which architectural framework, if any, is the best for any given situation. The problem is that one size does not fit all. Many standards are large, complex, and difficult to implement. They often do not address everyday situations your modeling teams face while trying to deliver complete, concise, and consistent models. Unfortunately, you are often locked into the paradigm provided the toolset and your teams begin to question the need for a framework in the first place.

A good modeling framework should be flexible enough to adapt to your particular environment. It should not dictate the use of specific terms, nor adamantly demand that you assemble components and relationships in a certain manner. However, once you have adapted the framework to your needs, it should provide the level of governance that you desire to place over your modeling teams. The framework should not only allow you to make changes to it as your experience grows, it should also automatically update your models to reflect those changes. Most importantly, a good framework should make your team's work easier to do, not harder.

An Analogy

To show the value of a good framework, let us look at an analogy from a completely different domain, photography.

Taking great pictures is both an art and a science. The underlying premise of photography comes down to the length of the exposure time (speed) and the size of the aperture opening. This seems simple enough. However, as anyone can tell you, taking great pictures requires an in-depth understanding of the subtleties of light and composition. The advent of cameras with automated metering helped the amateur take good photographs under average conditions. However, those cameras and the photo enthusiasts holding them often did not understand that, for example, a scene consisting of white snow has to be overexposed or the snow will appear a dingy gray in the resulting picture.

The next evolutionary step in photography was "smart cameras" programmed to adjust the exposure time and aperture according to the type of scene you are shooting. Set the camera on "Snow" and it will automatically overexpose the photograph by the correct amount. Set it for "Backlight", "Sports", or "Night" and the camera sets the exposure and speed accordingly, even turning on the flash if needed. The camera manufacturers have embedded the knowledge of the experts regarding the different scenes into the camera. Now, the amateur can simply select the appropriate scene, point, and shoot, and get a perfectly exposed photo. Of course, the photographer is still responsible for the composure of the picture, hopefully not cutting off Aunt Janet's head! Instead of dealing with time and aperture settings, the world of formal modeling deals with the much more complex concepts that UML has been designed to address. The decision process for the different "scenes" in modeling, therefore, is much more difficult than in our photography analogy. This makes the need for "smart modeling environments" all the more important.

Introducing the Model Guardian

OAD Consulting's new product, the Model Guardian framework & model management system for Enterprise Architect does for modeling what smart cameras do for photography. It embeds the knowledge of experts into a modeling framework that will let all of your modelers design like experts. Like the smart cameras, pre-defined "scenes" such as Enterprise Architecture, Enterprise Risk Management, Business Modeling, SOA, etc., and their modeling solutions are available. However, unlike the smart cameras, you are not limited to the pre-defined scenes. You can create your own or modify the pre-defined ones. Just as important, you can then update your existing models with the framework changes. Imagine if you could update your old pictures with the new features of the latest camera.

Model Guardian is designed to allow you to create the modeling environment you want without worrying about how to implement it in Enterprise Architect (EA). It goes beyond the capabilities inherent in EA by adding the following...

- A user interface that leads you through the process of creating and maintaining your modeling framework
- Automatic tracking of changes from one version of the framework to another with systemic application of those changes to your models
- Automatic incorporation of existing models into the framework & model management
- Automatic generation of SQL views that can be used directly within Enterprise Architect as model searches, with the RTF report writer, and with SQL report writers[1]
- User-defined views that are modeled visually within Enterprise Architect that leverage the auto-generated views and are automatically updated with each new version of the framework
- Pre-defined, extensible frameworks for different modeling scenarios[2]

The benefits derived from creating models using *Model Guardian* include…

- No need for an in-depth understanding of UML

[1] Several views are generated for each toolbox element and connector defined in the framework. The various views select the tagged values, exposed interfaces, embedded parts, relationships, etc. They also provide capabilities like selecting elements and/or connectors that appear on a given diagram or set of diagrams.
[2] OAD Consulting's (EA)^2 Enterprise Architecture Modeling Framework is one example of a pre-defined framework that works with *Model Guardian*.

The decision to use one UML construct vs. another is made by your experts when creating the framework. The framework then “coaches” modelers using the constructs you have built into the framework.

- Consistency in designs

  Using the framework ensures that model elements and connectors are drawn properly and receive the necessary tagged values. The framework relationships help ensure that the elements are properly related to each other. The designs created by different modelers are consistent, making the reports more effective. The framework makes modeling to your architectural standards the easiest way to go.

- Focused framework areas for different audiences

  While your underlying metamodel applies to the entire framework, you can create areas that are specific to certain audiences. For example, you can create toolboxes and diagram types specific to business modeling and others for application design. This means that each stakeholder has the view appropriate for his or her role. In addition, you can create domain specific model frameworks for the different business areas of your organization, e.g. member, plan sponsor, and claims for an insurance company.

- Model Validation

  It is important to understand how modelers are using the framework and to report on its inconsistent use. With *Model Guardian*, you can run reports to show where modelers have “gone outside” the framework and then evaluate whether this behavior represents infractions to the architectural rules you have established or whether they represent needed extensions to the framework.

- Comprehensive change management

  Change is inevitable and, therefore, framework evolution is one of the most important aspects of the success of a modeling program. *Model Guardian* allows you to make broad changes to the framework, adding new and renaming existing element types, connector types, tagged value definitions, relationships, toolboxes, and diagram types. It then automatically updates your models with the latest version of the framework.

Framework Editor and Model Editor

There are two editions to *Model Guardian*, the Framework Editor Edition and the Model Editor Edition. The former is used to create and maintain your modeling frameworks and the latter interprets them as you create and maintain your models. The Framework Editor includes all of the functionality of the Model Editor, so you can immediately test the changes you make to the framework. A typical deployment would have the Framework Editor for each person responsible for creating and maintaining your frameworks and the less expensive Model Editor for each modeler responsible for using the framework to develop your models. Those without either can still work with the resulting models and have full access to everything created in them. However, without the Framework Editor or Model Editor, they will not have the toolboxes, QuickLinks, automated tagged values, and validation capabilities.

The Framework Editor allows you to build the various elements that comprise a Modeling Framework in an incremental manner. You can start from scratch or use one of OAD's pre-developed frameworks and refine it to fit your needs. As you create the various elements of your framework, you can select the rules to be applied when creating models from the framework. For example, you can define the cardinality of tagged values that may be added to an object or connector, such as assigning only one "Business Lead" to a "Business Application". You may define relationships consisting of two endpoint object types and a connector type. To save on maintenance, you can create relationships between object types and have the relationships apply to all combinations of their subtypes.

Once you have created your framework, you can generate the technology file for use with Enterprise Architect and deploy it to users who have the Model Editor.

Framework Management Workflow

This section describes the "macro" process of creating and managing a framework and assumes you have a cursory knowledge of Enterprise Architect's user interface. Although the steps are listed in order, there is no prescribed order. As you add elements on one tab, they become available on the other tabs. This lets you to build your framework incrementally.

- Create Tag Definitions to be added to model elements and connectors
- Create a hierarchy of Metatypes, i.e. Object Types and Connector Types, assigning them either free form tags or tags from the Tag Definition tab. Metatypes inherit their parent types' tags and relationships.
- Create Relationships that consist of two endpoints, i.e. two Object Types, and a Connector Type. The Object Types and Connector Type can be from the framework or simply UML types. Using UML types allows you to create relationships to elements that are not associated with the framework.
- Create Toolbox Sections by adding appropriate Object Types and Connector Types and then ordering them.
- Create Toolboxes by adding the appropriate Toolbox Sections and then ordering them.
- Create Diagram Types, indicating the Toolbox to be associated with them.
- Create Modeling Domains by selecting the appropriate Diagram Types along with the Object and Connector Types that are on the Diagram Types Toolboxes.
- Save the framework. This can be done at any time as it simply saves the framework specifications to disk.
- Generate the EA technology file. This file will be used the next time Enterprise Architect is started, thus incorporating your framework.
- Generate the SQL views from the framework elements, including any user-defined views that have been modeled in EA.
- Test the framework in EA, exercising the toolboxes, Quicklinks, and trying the SQL views in model searches.
- Test the SQL views in your SQL reporting environment.
- Repeat the above steps, in any order, until you are ready to publish the framework for use by others.
- Publish the framework, which
    - increments the framework version number

  - saves the framework to disk
  - generates the technology file for the selected domains
  - adds new and changed elements to the framework history file
- Distribute the domain specific technology file to the appropriate users of the Model Editor.
- Using the Model Editor built into the Framework Editor or a stand-alone Model Editor, you can…
  - Update each model created using an earlier version of the framework with the new changes.
  - Install the updated SQL views into the SQL based EA repository.
  - Perform model-wide stereotype name changes, optionally populating the model elements with the framework elements, thus bringing them under framework management.
  - Perform model-wide tag name changes.

The following sections describe the elements of the framework and the steps to be taken.

Framework Elements

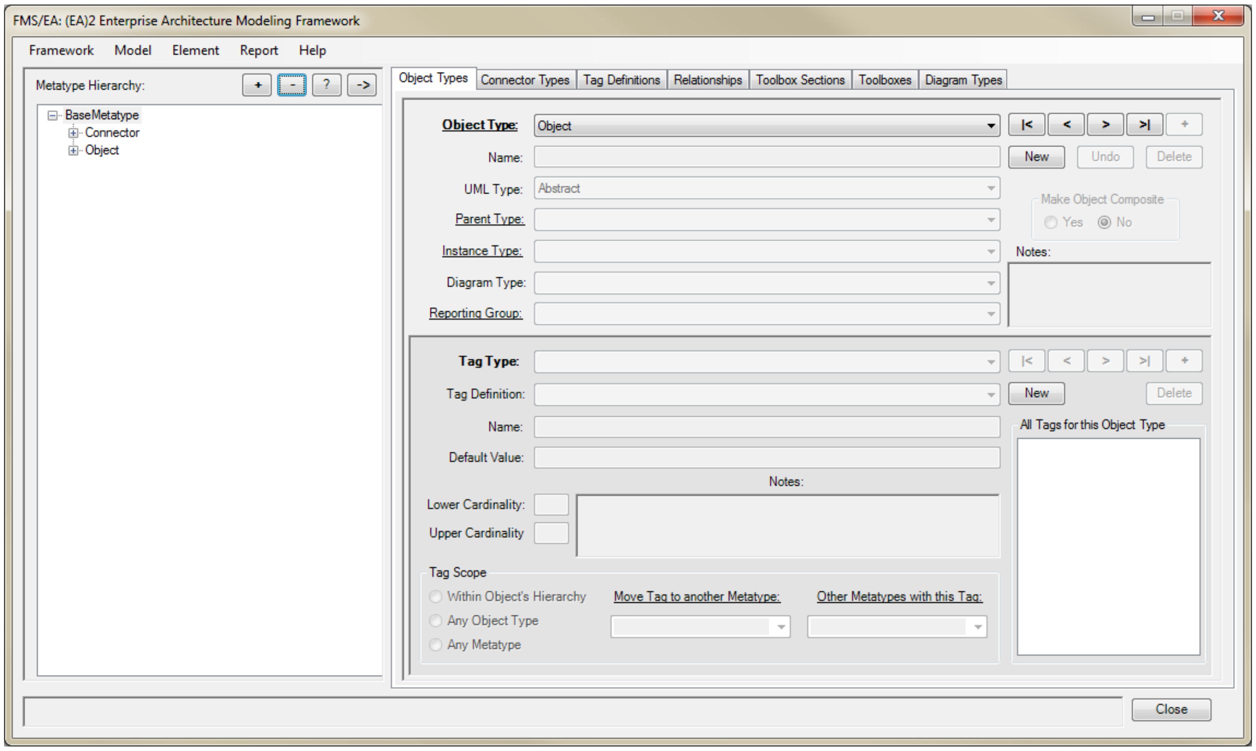
A Modeling Framework consists of the following elements:

- Tag Definitions
- Metatypes, i.e. Object Types and Connector Types
- Relationships
- Toolboxes and Toolbox Sections
- Diagram Types

Object Types and Connector Types

With the *Model Guardian* Framework Editor, you can create a hierarchy of Meta types that provide the EA toolbox tools and QuickLinks and much more. There are two branches in the hierarchy, one for Object Types and one for Connector Types. Both types can be assigned to EA toolboxes and used to define Relationships that can be manifested as EA QuickLinks.

The starting point for an empty modeling framework is shown below.

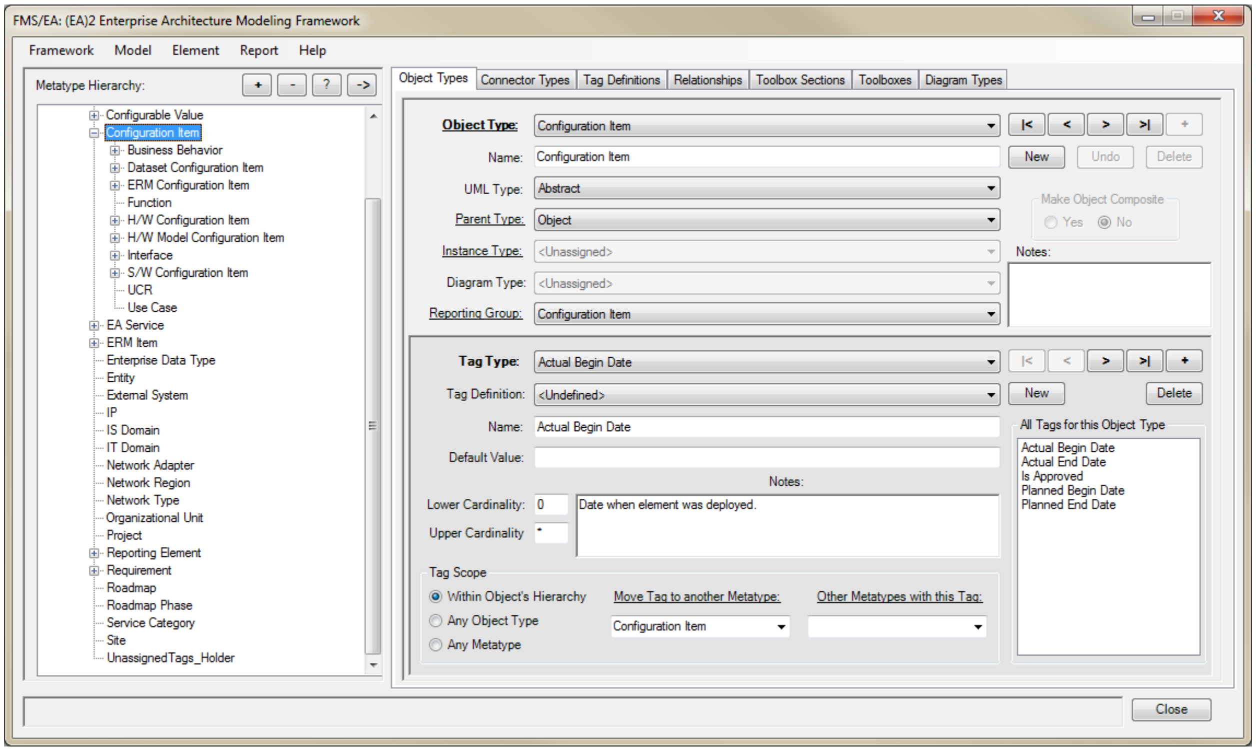


Object Type: Object

UML Type: Abstract

Make Object Composite: Yes [✓ No]

You build your type hierarchy by deriving new Meta types from the Object Types and Connector Types. The following image shows a screen shot of the Object Types tab with the hierarchy from the (EA)^2 Enterprise Architecture Modeling Framework:



Metatype Hierarchy:

- Configurable Value
- Configuration Item
  - Business Behavior
  - Dataset Configuration Item
  - ERM Configuration Item
  - Function
  - H/W Configuration Item
  - H/W Model Configuration Item
  - Interface
  - S/W Configuration Item
  - UCR
  - Use Case
- EA Service
- ERM Item
- Enterprise Data Type
- Entity
- External System
- IP
- IS Domain
- IT Domain
- Network Adapter
- Network Region
- Network Type
- Organizational Unit
- Project
- Reporting Element
- Requirement
- Roadmap
- Roadmap Phase
- Service Category
- Site
- UnassignedTags_Holder

Object Type: Configuration Item

Name: Configuration Item

UML Type: Abstract

Parent Type: Object

Instance Type: <Unassigned>

Diagram Type: <Unassigned>

Reporting Group: Configuration Item

Make Object Composite: Yes [✓ No]

Notes:

Tag Type: Actual Begin Date

Tag Definition: <Undefined>

Name: Actual Begin Date

Default Value:

Lower Cardinality: 0

Upper Cardinality: *

Notes: Date when element was deployed.

Tag Scope: [✓ Within Object's Hierarchy] Any Object Type Any Metatype

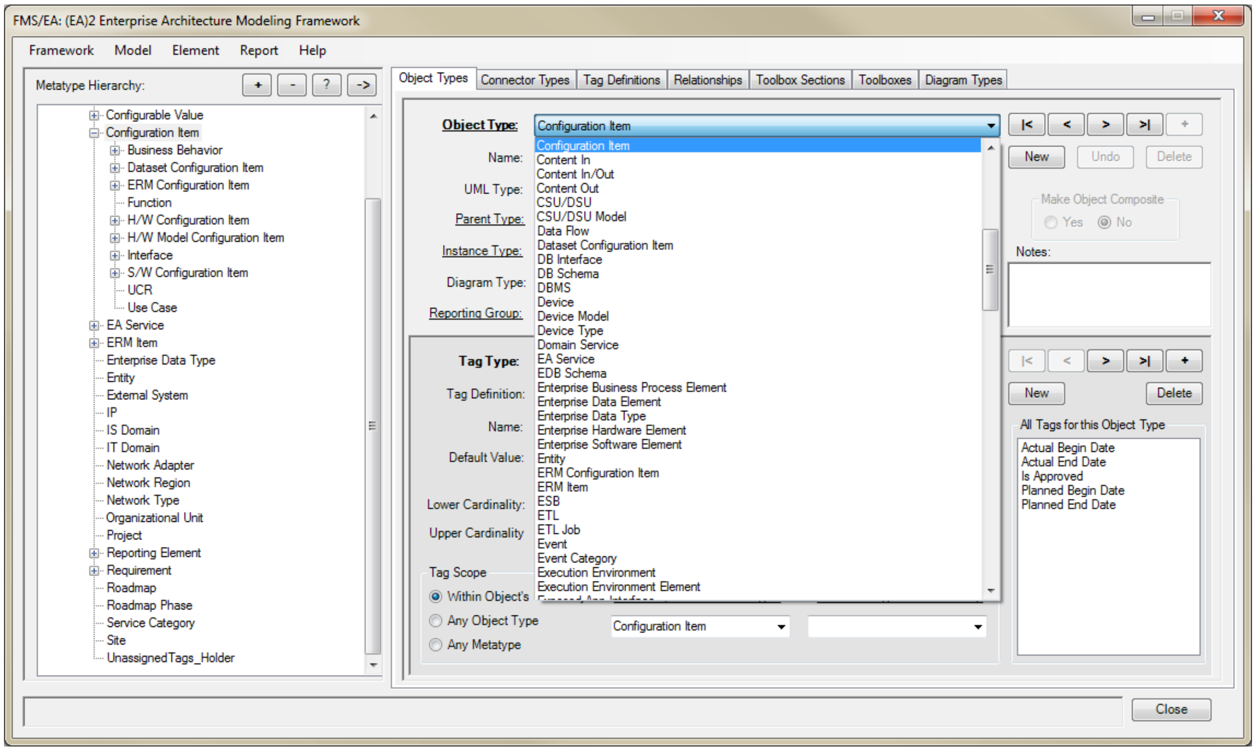
Move Tag to another Metatype: Configuration Item

Other Metatypes with this Tag:

All Tags for this Object Type:

Actual Begin Date
Actual End Date
Is Approved
Planned Begin Date
Planned End Date

You can select a Meta type in the hierarchy tree on the left and then move it to the editing area on the right. The main drop down combo box at the top of each window tab, e.g. the Object Type combo box in the image above, provides an alphabetized list of the types on that tab, as shown below.



Metatype Hierarchy:
- Configurable Value
- Configuration Item
  - Business Behavior
  - Dataset Configuration Item
  - ERM Configuration Item
  - Function
  - H/W Configuration Item
  - H/W Model Configuration Item
  - Interface
  - S/W Configuration Item
  - UCR
  - Use Case
- EA Service
- ERM Item
- Enterprise Data Type
- Entity
- External System
- IP
- IS Domain
- IT Domain
- Network Adapter
- Network Region
- Network Type
- Organizational Unit
- Project
- Reporting Element
- Requirement
- Roadmap
- Roadmap Phase
- Service Category
- Site
- UnassignedTags_Holder

Object Type: Configuration Item

Name:
UML Type:
Parent Type:
Instance Type:
Diagram Type:
Reporting Group:

Configuration Item
Content In
Content In/Out
Content Out
CSU/DSU
CSU/DSU Model
Data Flow
Dataset Configuration Item
DB Interface
DB Schema
DBMS
Device
Device Model
Device Type
Domain Service
EA Service
EDB Schema
Enterprise Business Process Element
Enterprise Data Element
Enterprise Data Type
Enterprise Hardware Element
Enterprise Software Element
Entity
ERM Configuration Item
ERM Item
ESB
ETL
ETL Job
Event
Event Category
Execution Environment
Execution Environment Element

Make Object Composite: Yes [✓ No]

Tag Type:
Tag Definition:
Name:
Default Value:
Lower Cardinality:
Upper Cardinality

Tag Scope: [✓ Within Object's] Any Object Type Any Metatype

Configuration Item

All Tags for this Object Type
- Actual Begin Date
- Actual End Date
- Is Approved
- Planned Begin Date
- Planned End Date

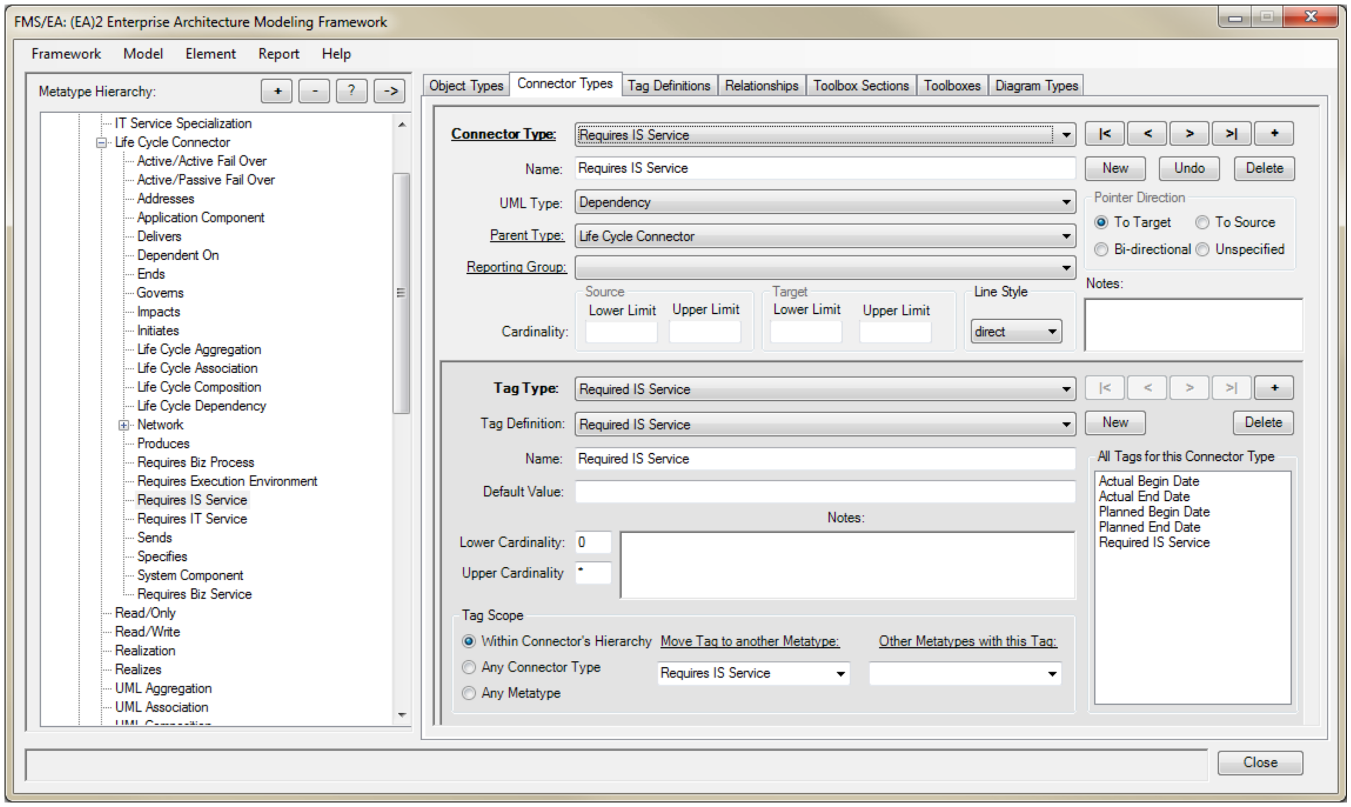
Whenever you select one, it is automatically located in the hierarchy tree on the left, so you can see exactly where it is within the hierarchy. You can also click on an underlined label, e.g. "Parent Type", to locate it in the hierarchy tree. When creating a new Meta type, its parent type will be the currently selected Meta type, but you can change its parent at any time, even in a later version of the framework.

You can mark an Object Type as "Composite", meaning that its main diagram will be opened when double clicking on the object. You can also set the Diagram Type with its associated Toolbox (see the section on Diagram Types below).

You may place tags at any level within the hierarchy. All Meta types derived from a Meta type with tags inherit those tags. They may also optionally inherit the relationships of its ancestral Meta types, as described below in the Relationships section.

The list box in the bottom right portion of the tab section shows all of the tags associated with the current Object Type, including those from the Object Type's ancestors. The section to the left of that labeled Tag Scope makes it easy to move a tag from one Object Type to another. The Tag Scope can be set to show 1) only those Object Types in the current Object Type's hierarchy, both above and below, 2) all Object types, or 3) all Metatypes, including both Object Types and Connector Types.

The next figure shows the Connector Types Tab:



FMS/EA: (EA)2 Enterprise Architecture Modeling Framework

Framework Model Element Report Help

Metatype Hierarchy:

- IT Service Specialization
- Life Cycle Connector
  - Active/Active Fail Over
  - Active/Passive Fail Over
  - Addresses
  - Application Component
  - Delivers
  - Dependent On
  - Ends
  - Governs
  - Impacts
  - Initiates
  - Life Cycle Aggregation
  - Life Cycle Association
  - Life Cycle Composition
  - Life Cycle Dependency
  - Network
  - Produces
  - Requires Biz Process
  - Requires Execution Environment
  - Requires IS Service
  - Requires IT Service
  - Sends
  - Specifies
  - System Component
  - Requires Biz Service
- Read/Only
- Read/Write
- Realization
- Realizes
- UML Aggregation
- UML Association

Object Types | Connector Types | Tag Definitions | Relationships | Toolbox Sections | Toolboxes | Diagram Types

Connector Type: Requires IS Service

Name: Requires IS Service

UML Type: Dependency

Parent Type: Life Cycle Connector

Reporting Group:

Pointer Direction: [✓ To Target] To Source Bi-directional Unspecified

Cardinality: Source Lower Limit, Upper Limit; Target Lower Limit, Upper Limit

Line Style: direct

Notes:

Tag Type: Required IS Service

Tag Definition: Required IS Service

Name: Required IS Service

Default Value:

Lower Cardinality: 0

Upper Cardinality: *

Notes:

All Tags for this Connector Type:
Actual Begin Date
Actual End Date
Planned Begin Date
Planned End Date
Required IS Service

Tag Scope: [✓ Within Connector's Hierarchy] Any Connector Type Any Metatype

Move Tag to another Metatype: Requires IS Service

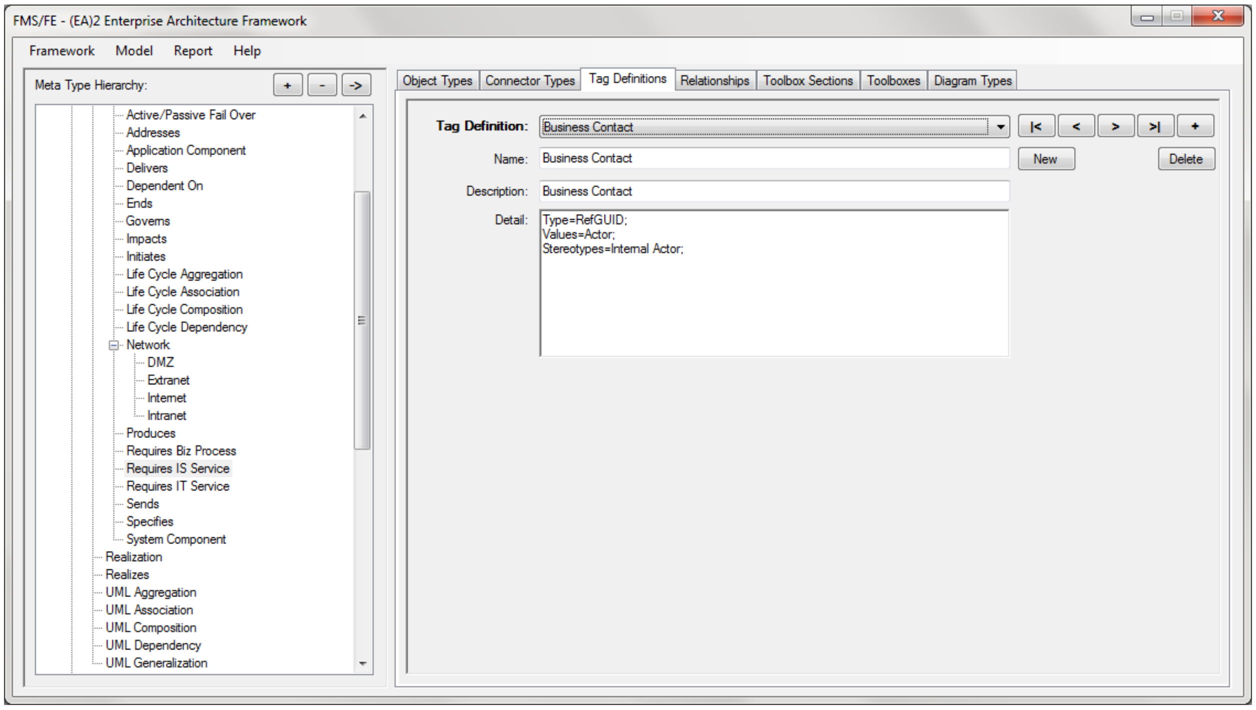
Other Metatypes with this Tag:

As you can see, both Object Types and Connector Types can have Tag Types associated with them. These can be ad hoc tags with free form entry or they can be Tag Definitions whose EA format has been defined, as you will see on the Tag Definitions Tab next. You can also set the line style, indicating how it will be drawn in EA, and the direction that the arrow should point, if any.

Tag Definitions

Tagged values can be added to model elements and connectors to provide additional information beyond what UML specifies. For example, the (EA)^2 Enterprise Architecture Framework automatically adds the Actual Begin Date, Actual End Date, Planned Begin Date, and Planned End Date to each model element or connector whose framework meta type is derived from the Configuration Item meta type. This provides information about the configuration items that is required to manage them but is not part of their design.

In EA, the tagged values can be unformatted strings or pre-defined types, e.g. a date format, an enumerated list, or a reference to a model element. You can define the pre-defined tags within *Model Guardian* as well and then assign them to the Object and Connector types, as shown in the next figure:



Tag Definition: Business Contact

Name: Business Contact

Description: Business Contact

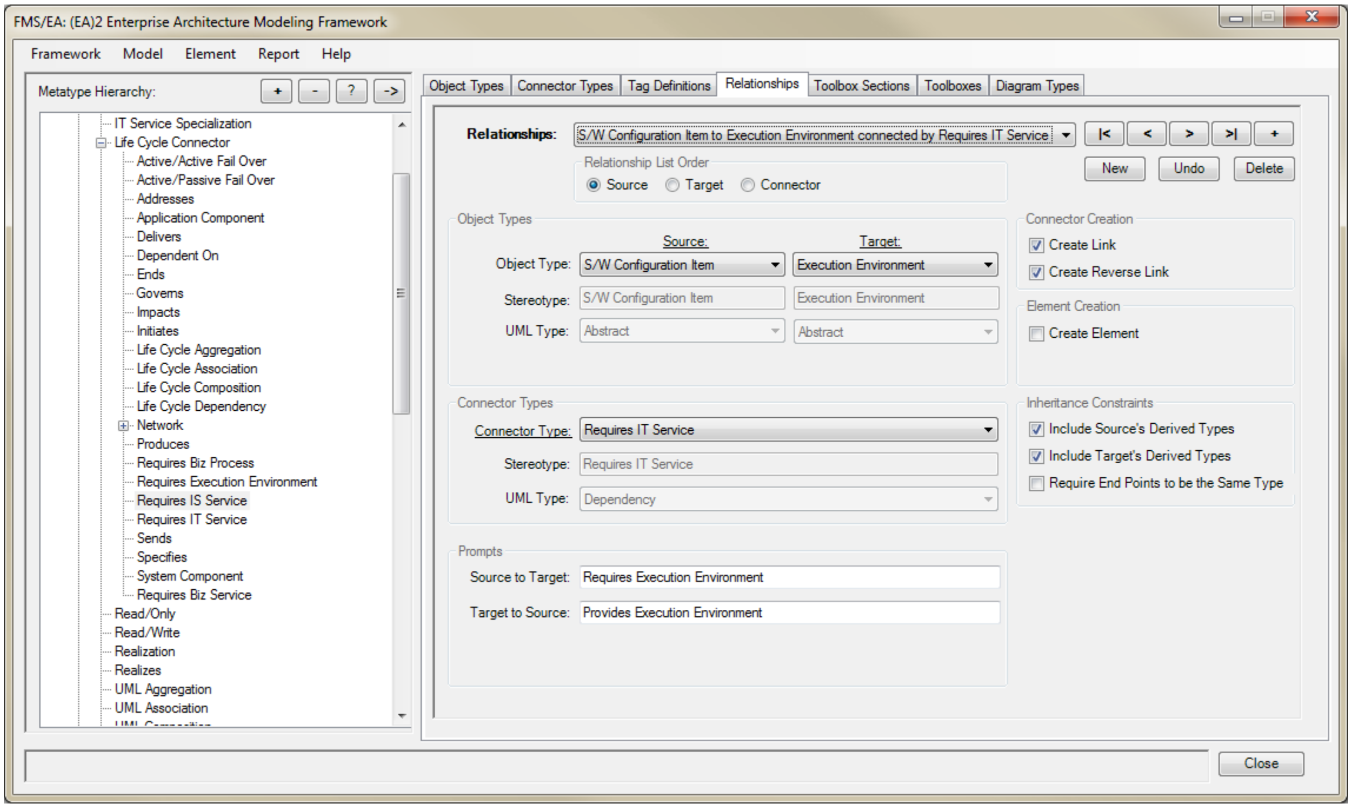
Detail:

```
Type=RefGUID;
Values=Actor;
Stereotypes=Internal Actor;
```

Just as you do in Enterprise Architect, you define the Tag's name, description, and the format. A RefGUID format is shown above which lets your tag link to an element within your model.

Relationships

Relationships define how your model elements can be linked together. They are composed of two Object Type endpoints and a Connector Type. The following image shows the Relationships Tab:


FMS/EA: (EA)2 Enterprise Architecture Modeling Framework

Framework Model Element Report Help

Metatype Hierarchy:

- IT Service Specialization
- Life Cycle Connector
  - Active/Active Fail Over
  - Active/Passive Fail Over
  - Addresses
  - Application Component
  - Delivers
  - Dependent On
  - Ends
  - Governs
  - Impacts
  - Initiates
  - Life Cycle Aggregation
  - Life Cycle Association
  - Life Cycle Composition
  - Life Cycle Dependency
  - Network
  - Produces
  - Requires Biz Process
  - Requires Execution Environment
  - Requires IS Service
  - Requires IT Service
  - Sends
  - Specifies
  - System Component
  - Requires Biz Service
- Read/Only
- Read/Write
- Realization
- Realizes
- UML Aggregation
- UML Association

Object Types | Connector Types | Tag Definitions | Relationships | Toolbox Sections | Toolboxes | Diagram Types

Relationships: S/W Configuration Item to Execution Environment connected by Requires IT Service

Relationship List Order: (●) Source ( ) Target ( ) Connector

Object Types

	Source:	Target:
Object Type:	S/W Configuration Item	Execution Environment
Stereotype:	S/W Configuration Item	Execution Environment
UML Type:	Abstract	Abstract

Connector Types

Connector Type: Requires IT Service

Stereotype: Requires IT Service

UML Type: Dependency

Prompts

Source to Target: Requires Execution Environment

Target to Source: Provides Execution Environment

Connector Creation

- ✓ Create Link
- ✓ Create Reverse Link

Element Creation

- ☐ Create Element

Inheritance Constraints

- ✓ Include Source's Derived Types
- ✓ Include Target's Derived Types
- ☐ Require End Points to be the Same Type

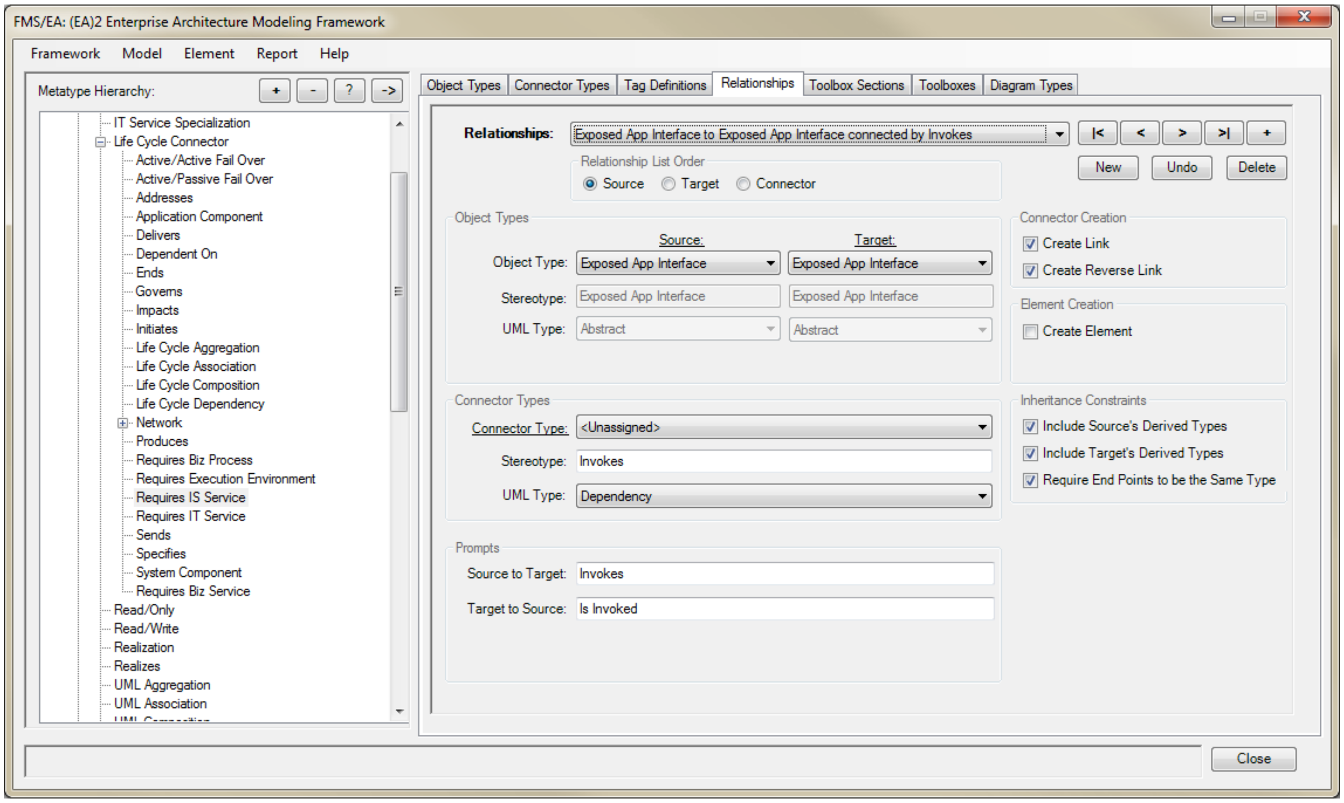
All three of the key elements can be meta types from the framework or simply UML types. In the image above, we see two Object Types, “S/W Configuration Item” and “Execution Environment”. Their stereotypes and UML types have already been set on the Object Types tab. You can leave these unassigned and set the enter the stereotype and the UML type. This lets you link elements that are not part of your framework. The same holds true for the Connector Type. However, whenever using an Object Type or Connector Type from the framework, your model elements will automatically receive all of the tagged values you have defined for them.

Relationships can be inherited. For example, in the image above, we can see that “Include Source’s Derived Types” and “Include Target’s Derived Types” are checked. The source, “S/W Configuration Item” and the target, “Execution Environment”, have several derived types as seen in the Meta Type Hierarchy tree. Therefore, QuickLinks and rule sets will be generated for relationships between each subtype of “S/W Configuration Item” and each subtype of “Execution Environment”. Any enforcement and reporting of infractions will also be provided for the derived types. This allows you to define dozens of relationships with a single entry in the form.

Other options include

- *Create Reverse Link* – checking this option results in a Quicklink being set up for dragging from the target back to the source.
- *Create Element* – checking this option means that the Target Object Type will be created when dragging the QuickLink arrow to empty space on a diagram.
- *Embed Element* – checking this option results in the Source Object Type being embedded in the Target Object, for example, when drawing from an exposed interface to a component.

- *Require End Points to be the Same* – this option, which is used when Quicklinks for child Object Types are being created, will establish QuickLinks when the child Source Object Type is the same as the Target Object Type. For example, in the image below, relationships are being set up for drawing between "Exposed App Interfaces"̲. Selecting this option means that a QuickLink will be set up for connecting an "Async" exposed interface to another "Async" exposed interface but not to the other types of "Exposed App Interfaces".
- *Prefix* – This allows you to automatically place a prefix in front of the drop down prompts for your QuickLinks, identifying those Relationships that you have defined within your framework.



Relationships: Exposed App Interface to Exposed App Interface connected by Invokes

Relationship List Order: [✓ Source] Target Connector

Object Types

	Source:	Target:
Object Type:	Exposed App Interface	Exposed App Interface
Stereotype:	Exposed App Interface	Exposed App Interface
UML Type:	Abstract	Abstract

Connector Types

Connector Type: <Unassigned>

Stereotype: Invokes

UML Type: Dependency

Prompts

Source to Target: Invokes

Target to Source: Is Invoked

Connector Creation

- ✓ Create Link
- ✓ Create Reverse Link

Element Creation

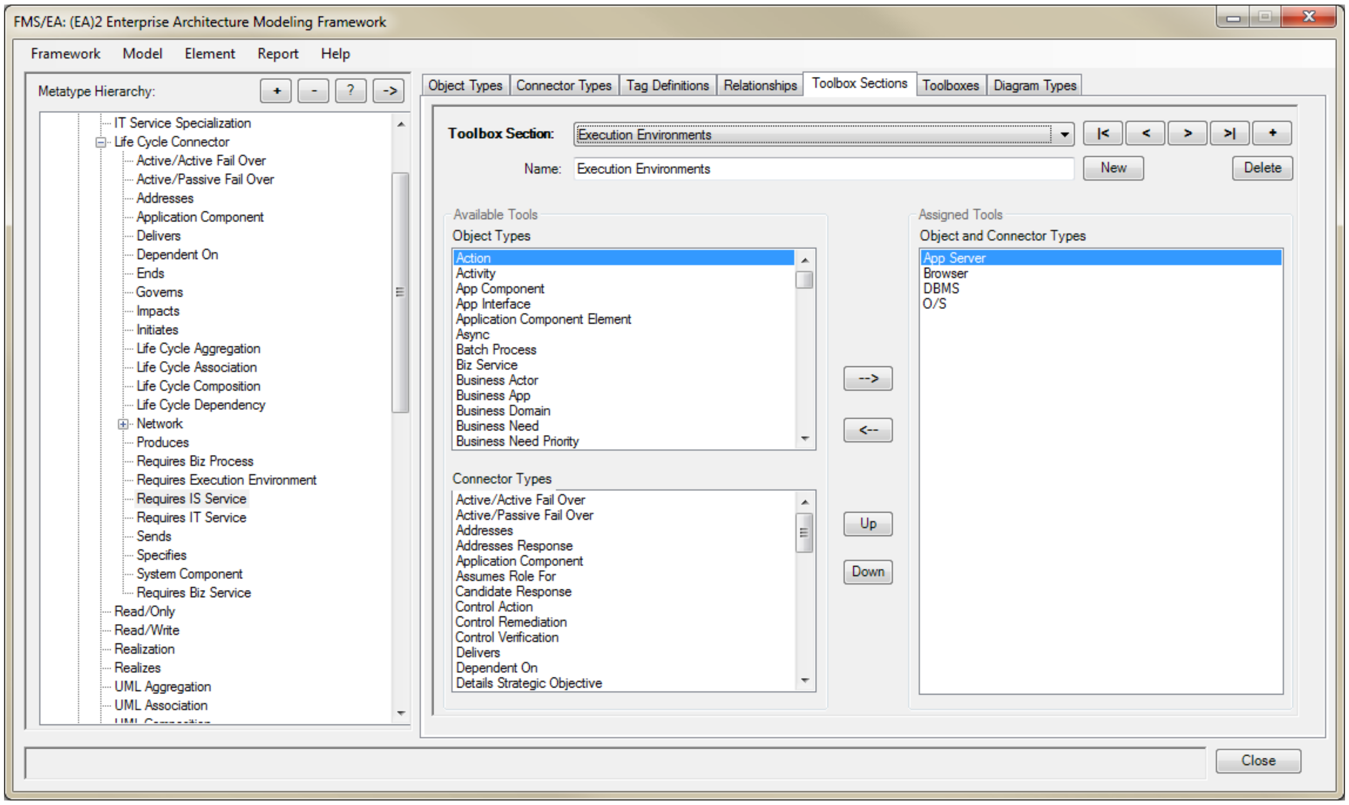
- ☐ Create Element

Inheritance Constraints

- ✓ Include Source's Derived Types
- ✓ Include Target's Derived Types
- ✓ Require End Points to be the Same Type

Toolboxes and Toolbox Sections

The various Meta types of the hierarchy can be grouped together into Toolbox Sections or Pages. These Toolbox Sections can then be added to the Toolboxes that you see when you open the different diagram types. The following image shows the Toolbox Sections Tab:


FMS/EA: (EA)2 Enterprise Architecture Modeling Framework

Framework Model Element Report Help

Metatype Hierarchy: + - ? ->

- IT Service Specialization
- Life Cycle Connector
  - Active/Active Fail Over
  - Active/Passive Fail Over
  - Addresses
  - Application Component
  - Delivers
  - Dependent On
  - Ends
  - Governs
  - Impacts
  - Initiates
  - Life Cycle Aggregation
  - Life Cycle Association
  - Life Cycle Composition
  - Life Cycle Dependency
  - Network
  - Produces
  - Requires Biz Process
  - Requires Execution Environment
  - Requires IS Service
  - Requires IT Service
  - Sends
  - Specifies
  - System Component
  - Requires Biz Service
- Read/Only
- Read/Write
- Realization
- Realizes
- UML Aggregation
- UML Association

Object Types | Connector Types | Tag Definitions | Relationships | Toolbox Sections | Toolboxes | Diagram Types

Toolbox Section: Execution Environments |< < > >| +

Name: Execution Environments New Delete

Available Tools

Object Types

Action
Activity
App Component
App Interface
Application Component Element
Async
Batch Process
Biz Service
Business Actor
Business App
Business Domain
Business Need
Business Need Priority

Connector Types

Active/Active Fail Over
Active/Passive Fail Over
Addresses
Addresses Response
Application Component
Assumes Role For
Candidate Response
Control Action
Control Remediation
Control Verification
Delivers
Dependent On
Details Strategic Objective

--> <-- Up Down

Assigned Tools

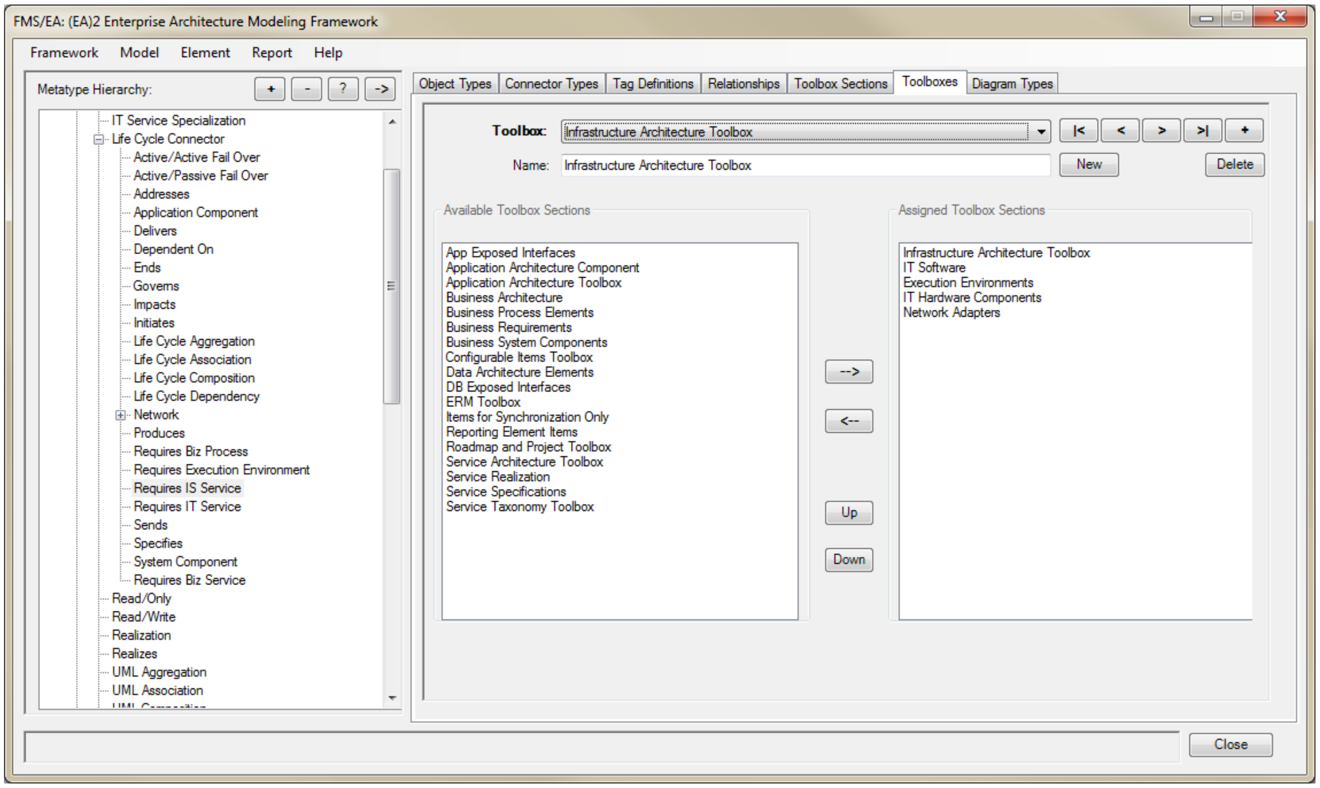
Object and Connector Types

App Server
Browser
DBMS
O/S

Close

After assigning tools to the Assigned Tools list, you can order them as you like.

The next image shows the Toolbox Tab:


FMS/EA: (EA)2 Enterprise Architecture Modeling Framework

Framework Model Element Report Help

Metatype Hierarchy: + - ? ->

- IT Service Specialization
- Life Cycle Connector
  - Active/Active Fail Over
  - Active/Passive Fail Over
  - Addresses
  - Application Component
  - Delivers
  - Dependent On
  - Ends
  - Governs
  - Impacts
  - Initiates
  - Life Cycle Aggregation
  - Life Cycle Association
  - Life Cycle Composition
  - Life Cycle Dependency
  - Network
  - Produces
  - Requires Biz Process
  - Requires Execution Environment
  - Requires IS Service
  - Requires IT Service
  - Sends
  - Specifies
  - System Component
  - Requires Biz Service
- Read/Only
- Read/Write
- Realization
- Realizes
- UML Aggregation
- UML Association

Object Types | Connector Types | Tag Definitions | Relationships | Toolbox Sections | Toolboxes | Diagram Types

Toolbox: Infrastructure Architecture Toolbox |< < > >| +

Name: Infrastructure Architecture Toolbox New Delete

Available Toolbox Sections

App Exposed Interfaces
Application Architecture Component
Application Architecture Toolbox
Business Architecture
Business Process Elements
Business Requirements
Business System Components
Configurable Items Toolbox
Data Architecture Elements
DB Exposed Interfaces
ERM Toolbox
Items for Synchronization Only
Reporting Element Items
Roadmap and Project Toolbox
Service Architecture Toolbox
Service Realization
Service Specifications
Service Taxonomy Toolbox

--> <-- Up Down

Assigned Toolbox Sections

Infrastructure Architecture Toolbox
IT Software
Execution Environments
IT Hardware Components
Network Adapters

Close

The Toolbox Sections can be reused on as many Toolboxes as you want.

The next image shows the resulting Toolbox that appears in Enterprise Architect, along with its Toolbox Sections:

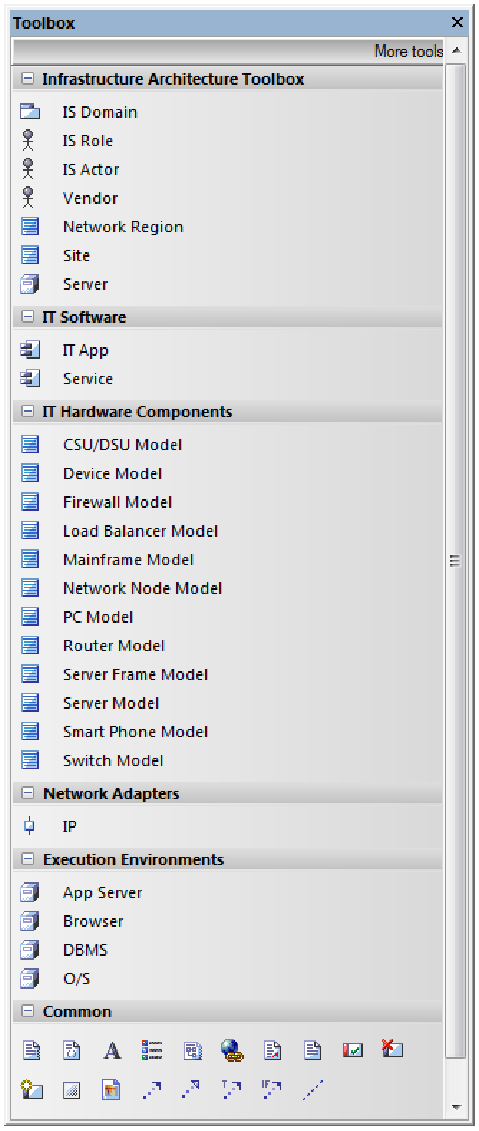
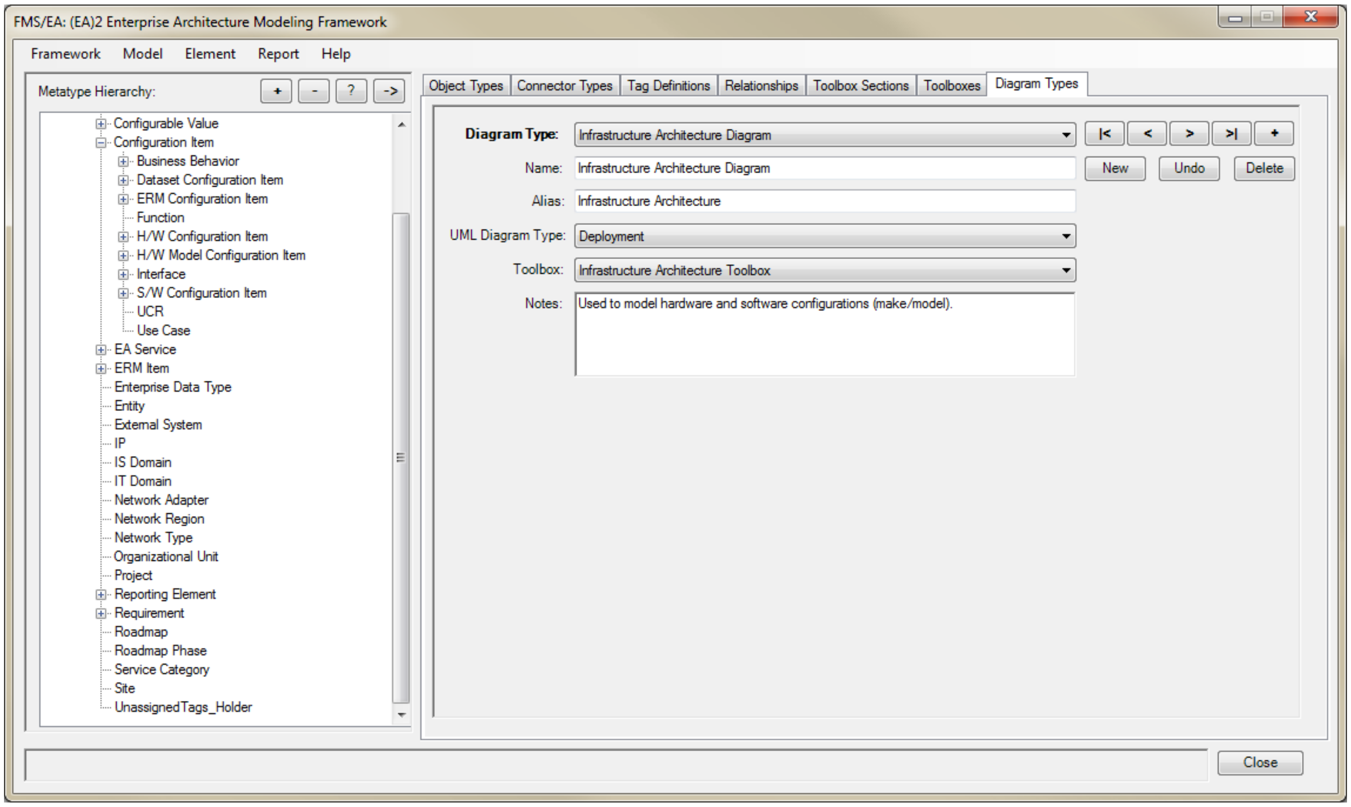


Diagram Types

The next step is to define the Diagram Types and their associated Toolboxes, as shown in the next image.



Configurable Value
Configuration Item
Business Behavior
Dataset Configuration Item
ERM Configuration Item
Function
H/W Configuration Item
H/W Model Configuration Item
Interface
S/W Configuration Item
UCR
Use Case
EA Service
ERM Item
Enterprise Data Type
Entity
External System
IP
IS Domain
IT Domain
Network Adapter
Network Region
Network Type
Organizational Unit
Project
Reporting Element
Requirement
Roadmap
Roadmap Phase
Service Category
Site
UnassignedTags_Holder

Diagram Type: Infrastructure Architecture Diagram
Name: Infrastructure Architecture Diagram
Alias: Infrastructure Architecture
UML Diagram Type: Deployment
Toolbox: Infrastructure Architecture Toolbox
Notes: Used to model hardware and software configurations (make/model).

Object Types that are set as Composite are given a Child Diagram Type so that you will get the appropriate Toolbox when opening the child diagram for the Object.

Menu Options

The following describes the menu options that are available in the Framework Editor. The Model Editor has all of the same options except the Framework options. The tabs are available for review in the Model Editor so that all of your modelers can see the “big picture” you have created within your framework.

Options

Save – This saves the framework specifications after validating and updating any changes on the tabs. If there are any unsaved changes when you exit Model Guardian, you will be asked whether you want to save them at that time. If you don’t, the changes still remain intact until you exit Enterprise Architect, at which point, you will be asked again. However, if you wait until you are exiting Enterprise Architect, you will not be able to correct any errors on the tabs and the changes will be lost.

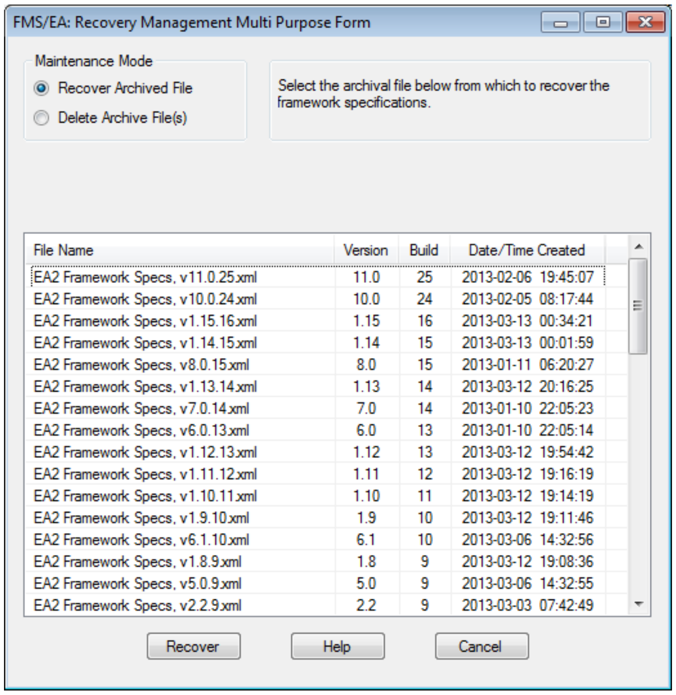
Generate – This first saves any framework changes and then, depending on which sub menu item is selected, generates the technology file for use with Enterprise Architect or the SQL view installation file for the underlying SQL DBMS. The generated files are placed in the Work-in-Progress folder. The changes will take effect the next time you start Enterprise Architect with the Model Guardian Framework Editor. This allows you to test the changes in a test environment prior to publishing them for general use. See the next section on publishing for information on generating the files for production use.

Publish – This saves any changes, versions the framework as an important step for maintaining the models created using the framework, and then generates the published version of the MDG Technology file and SQL View Installation file and places them in the Deployment folder. You can then distribute the files in that folder to Deployment folder on the computers with either the Framework Editor Edition or the Model Editor Edition. You can find where these folders are located by looking at the Model Guardian 'Help | About' menu option.

Recover Archived Version – Each time the framework is saved, the previous version is archived. This option presents the form below on the left, which allows you to retrieve a previous version of the framework, inspect it, and then decide whether to keep it or revert to the most recent version again. By clicking on the 'Delete Archived File(s)' option box, the form switches function as seen on the right. Now you can delete older versions with the help of the selection buttons:

- All : this selects all of the archived files, letting you uncheck those you want to keep.
- None : unselects all of the files.
- All Unpublished Versions : this checks all of the versions that do not end with a dot zero, i.e. all of the incremental builds you saved before you published a new version. In other words, it will keep all of the versions that were published and distributed, deleting the rest.
- All But the Last 5 Unpublished Versions – this leaves on the last 5 published versions unchecked.

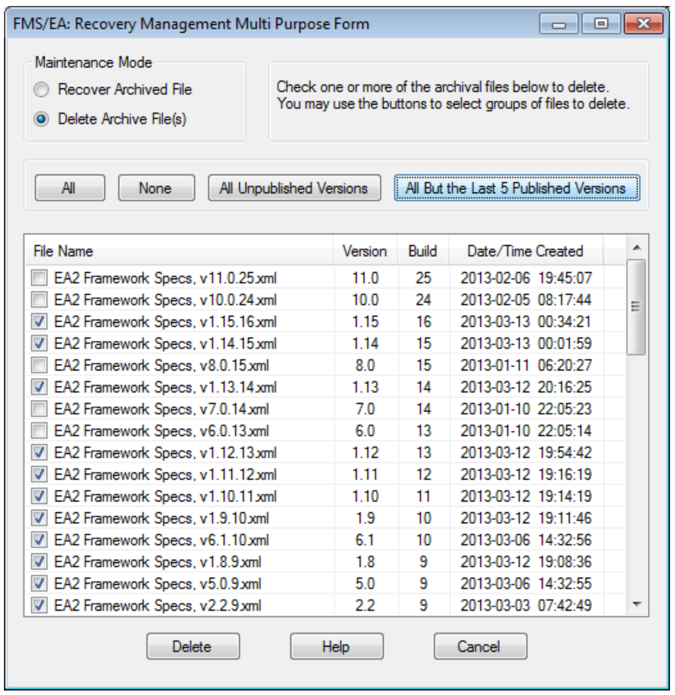
Once you have selected which files you want to delete, you can then delete them or cancel out of the operation by using the buttons at the bottom of the form.



Maintenance Mode: [● Recover Archived File] ○ Delete Archive File(s)

Select the archival file below from which to recover the framework specifications.

File Name	Version	Build	Date/Time Created
EA2 Framework Specs, v11.0.25.xml	11.0	25	2013-02-06 19:45:07
EA2 Framework Specs, v10.0.24.xml	10.0	24	2013-02-05 08:17:44
EA2 Framework Specs, v1.15.16.xml	1.15	16	2013-03-13 00:34:21
EA2 Framework Specs, v1.14.15.xml	1.14	15	2013-03-13 00:01:59
EA2 Framework Specs, v8.0.15.xml	8.0	15	2013-01-11 06:20:27
EA2 Framework Specs, v1.13.14.xml	1.13	14	2013-03-12 20:16:25
EA2 Framework Specs, v7.0.14.xml	7.0	14	2013-01-10 22:05:23
EA2 Framework Specs, v6.0.13.xml	6.0	13	2013-01-10 22:05:14
EA2 Framework Specs, v1.12.13.xml	1.12	13	2013-03-12 19:54:42
EA2 Framework Specs, v1.11.12.xml	1.11	12	2013-03-12 19:16:19
EA2 Framework Specs, v1.10.11.xml	1.10	11	2013-03-12 19:14:19
EA2 Framework Specs, v1.9.10.xml	1.9	10	2013-03-12 19:11:46
EA2 Framework Specs, v6.1.10.xml	6.1	10	2013-03-06 14:32:56
EA2 Framework Specs, v1.8.9.xml	1.8	9	2013-03-12 19:08:36
EA2 Framework Specs, v5.0.9.xml	5.0	9	2013-03-06 14:32:55
EA2 Framework Specs, v2.2.9.xml	2.2	9	2013-03-03 07:42:49



Maintenance Mode: ○ Recover Archived File [● Delete Archive File(s)]

Check one or more of the archival files below to delete.
You may use the buttons to select groups of files to delete.

	File Name	Version	Build	Date/Time Created
☐	EA2 Framework Specs, v11.0.25.xml	11.0	25	2013-02-06 19:45:07
☐	EA2 Framework Specs, v10.0.24.xml	10.0	24	2013-02-05 08:17:44
✓	EA2 Framework Specs, v1.15.16.xml	1.15	16	2013-03-13 00:34:21
✓	EA2 Framework Specs, v1.14.15.xml	1.14	15	2013-03-13 00:01:59
☐	EA2 Framework Specs, v8.0.15.xml	8.0	15	2013-01-11 06:20:27
✓	EA2 Framework Specs, v1.13.14.xml	1.13	14	2013-03-12 20:16:25
☐	EA2 Framework Specs, v7.0.14.xml	7.0	14	2013-01-10 22:05:23
☐	EA2 Framework Specs, v6.0.13.xml	6.0	13	2013-01-10 22:05:14
✓	EA2 Framework Specs, v1.12.13.xml	1.12	13	2013-03-12 19:54:42
✓	EA2 Framework Specs, v1.11.12.xml	1.11	12	2013-03-12 19:16:19
✓	EA2 Framework Specs, v1.10.11.xml	1.10	11	2013-03-12 19:14:19
✓	EA2 Framework Specs, v1.9.10.xml	1.9	10	2013-03-12 19:11:46
✓	EA2 Framework Specs, v6.1.10.xml	6.1	10	2013-03-06 14:32:56
✓	EA2 Framework Specs, v1.8.9.xml	1.8	9	2013-03-12 19:08:36
✓	EA2 Framework Specs, v5.0.9.xml	5.0	9	2013-03-06 14:32:55
✓	EA2 Framework Specs, v2.2.9.xml	2.2	9	2013-03-03 07:42:49

Merge (EA)^2 – for users who are converting from professional editions of the (EA)^2 Enterprise Architecture Framework and have made changes to its meta model.

Model Options

Apply Framework Changes – Synchronizes the elements and connectors of the currently opened model to the latest version of the framework.

Install SQL Views – Installs the generated SQL views into the EA repository running on SQL Server (support for others DBMS will be available soon).

Reports

Framework reports print out the structure of your framework for review. Validation reports can be generated to find out where modelers have gone outside the framework. You can also report on problems with a model after it has been updated to the latest version of your framework. For example, the report can flag elements created from a Meta type that was in a previous version of the framework but deleted in a later version, so you can decide what to do with these "orphans".

Framework Structure Report (Hierarchical View) – Prints a hierarchical view of the Meta types in your framework. This is the same view as in the hierarchy tree.

Framework Structure Report (Alphabetical View) – Prints an alphabetical list of the Meta types in your framework. This is the same view as in the combo boxes.

Model Validation Report – Prints a report showing model exceptions to the framework. For example, it will optionally show all of the relationships in the model that are not defined in the framework. It will also show elements that have an incomplete set of tagged values or have the wrong number of tagged values for those tags whose definition provides cardinality.

Versioning and Model Updates

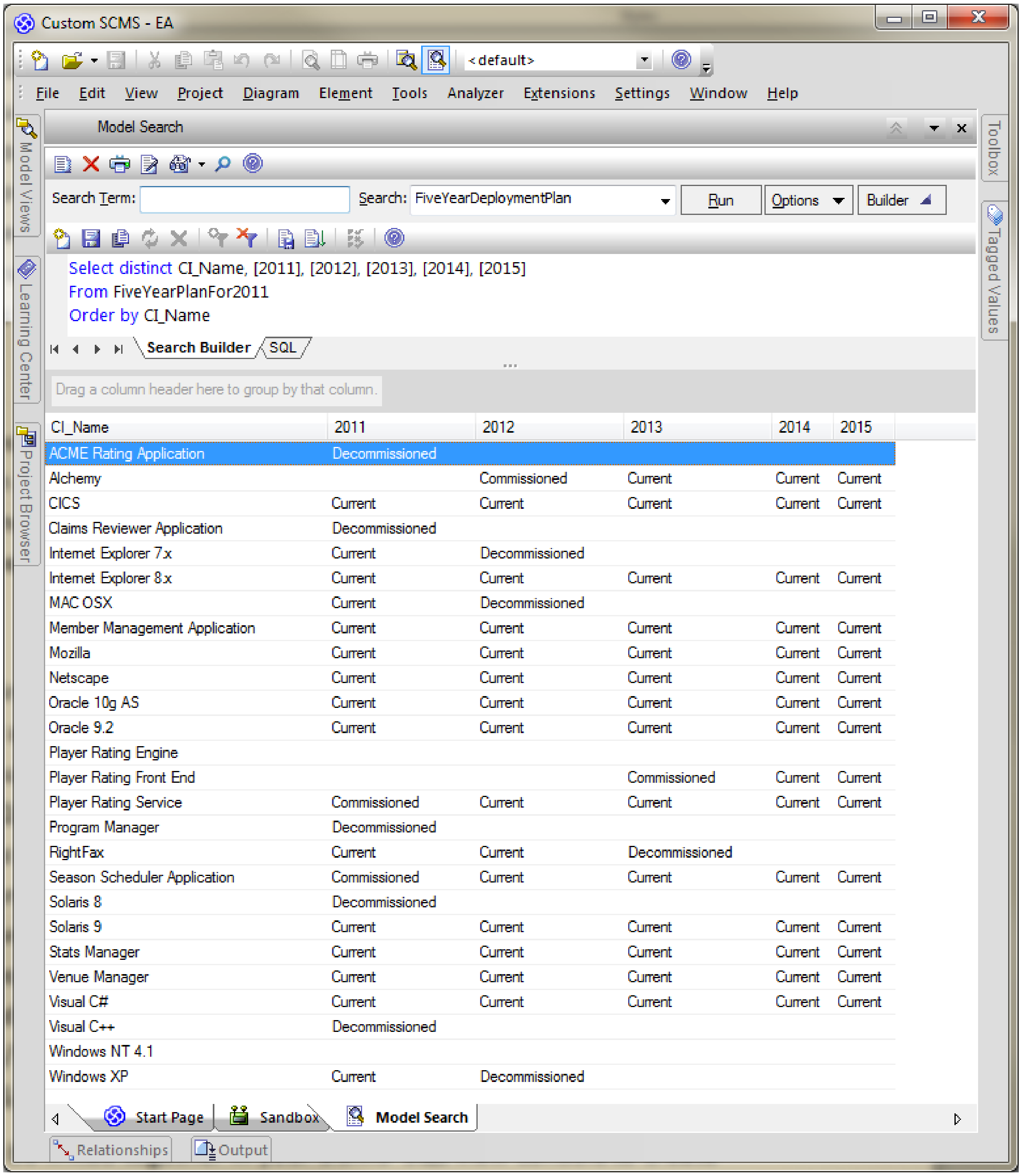
You will never get things right the first time, so *Model Guardian* keeps track of the elements as you make changes. There is a convenient Undo feature that lets you back out changes to any element until you have saved the framework. Once you are satisfied with the framework, you can version it, which automatically archives the previous version. The elements and connectors in each model that you create using the Model Editor are stamped with the version of the framework that was used to create them. This allows updating a model created using a previous version of the framework to the current version a snap. You will no longer need to synchronize each of the toolbox elements, one stereotype at a time after you have made changes to your framework.

SQL Views

Those familiar with the $(EA)^2$ Enterprise Architecture Modeling Framework, Reporting Editions are familiar with the powerful SQL views that allow you to create complex reports based on your EA repository. *Model Guardian* puts you in full control of the views. Once you are satisfied with your framework, the *Model Guardian* Framework Editor will generate SQL views for all of your Meta types. Each Meta type will have a basic and numerous supplementary views to be used for different scenarios. For example, suppose you have a Business App derived from S/W Configuration Item derived from Configuration Item derived from Object. Each level in the hierarchy will have a basic view created for it plus several supplementary views, e.g. the object along with its tagged values, the object with all of its relationships to other objects, the diagrams on which the object appears, and more. The views at each level include the Object Types inherited from it. Thus, using one of the S/W Configuration Item views will include all of the Business Apps and IT Apps and any other subtypes of the S/W Configuration Item type.

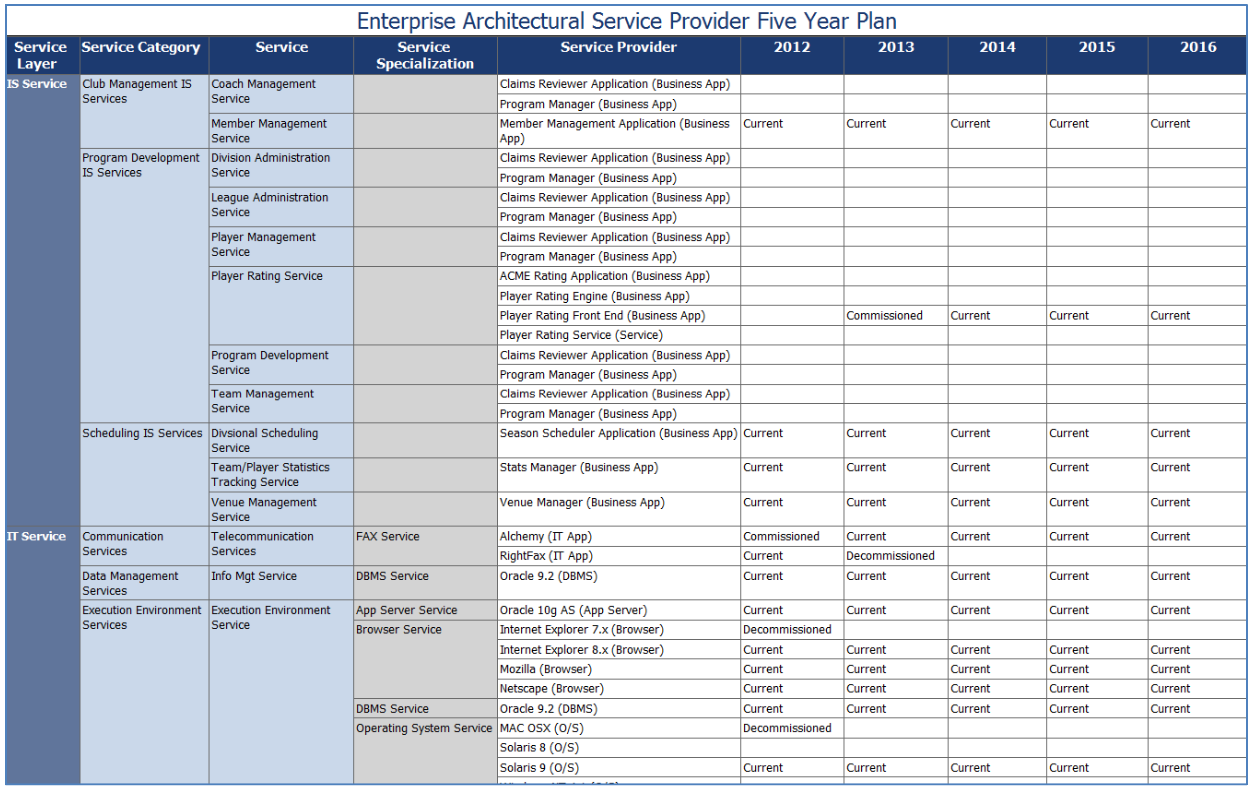
The views provide building blocks that can be linked together diagrammatically in EA to create user-defined views like the $(EA)^2$ Enterprise Architecture Modeling Framework's "Five Year Deployment Plan" for your configuration items. The views can then be used not just with your SQL report tool of choice, but they are also available within EA. The next image shows EA using the FiveYearDeploymentPlan and the following one shows a more extensive report generated by the Microsoft SQL Server Business Intelligence report writer.

Model Guardian



```
Select distinct CI_Name, [2011], [2012], [2013], [2014], [2015]
From FiveYearPlanFor2011
Order by CI_Name
```

CI_Name	2011	2012	2013	2014	2015
ACME Rating Application	Decommissioned				
Alchemy		Commissioned	Current	Current	Current
CICS	Current	Current	Current	Current	Current
Claims Reviewer Application	Decommissioned				
Internet Explorer 7.x	Current	Decommissioned			
Internet Explorer 8.x	Current	Current	Current	Current	Current
MAC OSX	Current	Decommissioned			
Member Management Application	Current	Current	Current	Current	Current
Mozilla	Current	Current	Current	Current	Current
Netscape	Current	Current	Current	Current	Current
Oracle 10g AS	Current	Current	Current	Current	Current
Oracle 9.2	Current	Current	Current	Current	Current
Player Rating Engine					
Player Rating Front End			Commissioned	Current	Current
Player Rating Service	Commissioned	Current	Current	Current	Current
Program Manager	Decommissioned				
RightFax	Current	Current	Decommissioned		
Season Scheduler Application	Commissioned	Current	Current	Current	Current
Solaris 8	Decommissioned				
Solaris 9	Current	Current	Current	Current	Current
Stats Manager	Current	Current	Current	Current	Current
Venue Manager	Current	Current	Current	Current	Current
Visual C#	Current	Current	Current	Current	Current
Visual C++	Decommissioned				
Windows NT 4.1					
Windows XP	Current	Decommissioned			



Enterprise Architectural Service Provider Five Year Plan

Service Layer	Service Category	Service	Service Specialization	Service Provider	2012	2013	2014	2015	2016
IS Service	Club Management IS Services	Coach Management Service		Claims Reviewer Application (Business App)					
				Program Manager (Business App)					
		Member Management Service		Member Management Application (Business App)	Current	Current	Current	Current	Current
	Program Development IS Services	Division Administration Service		Claims Reviewer Application (Business App)					
				Program Manager (Business App)					
		League Administration Service		Claims Reviewer Application (Business App)					
				Program Manager (Business App)					
		Player Management Service		Claims Reviewer Application (Business App)					
				Program Manager (Business App)					
		Player Rating Service		ACME Rating Application (Business App)					
				Player Rating Engine (Business App)					
				Player Rating Front End (Business App)		Commissioned	Current	Current	Current
				Player Rating Service (Service)					
		Program Development Service		Claims Reviewer Application (Business App)					
				Program Manager (Business App)					
		Team Management Service		Claims Reviewer Application (Business App)					
				Program Manager (Business App)					
	Scheduling IS Services	Divsional Scheduling Service		Season Scheduler Application (Business App)	Current	Current	Current	Current	Current
		Team/Player Statistics Tracking Service		Stats Manager (Business App)	Current	Current	Current	Current	Current
		Venue Management Service		Venue Manager (Business App)	Current	Current	Current	Current	Current
IT Service	Communication Services	Telecommunication Services	FAX Service	Alchemy (IT App)	Commissioned	Current	Current	Current	Current
				RightFax (IT App)	Current	Decommissioned			
	Data Management Services	Info Mgt Service	DBMS Service	Oracle 9.2 (DBMS)	Current	Current	Current	Current	Current
	Execution Environment Services	Execution Environment Service	App Server Service	Oracle 10g AS (App Server)	Current	Current	Current	Current	Current
			Browser Service	Internet Explorer 7.x (Browser)	Decommissioned				
				Internet Explorer 8.x (Browser)	Current	Current	Current	Current	Current
				Mozilla (Browser)	Current	Current	Current	Current	Current
				Netscape (Browser)	Current	Current	Current	Current	Current
			DBMS Service	Oracle 9.2 (DBMS)	Current	Current	Current	Current	Current
			Operating System Service	MAC OSX (O/S)	Decommissioned				
				Solaris 8 (O/S)					
				Solaris 9 (O/S)	Current	Current	Current	Current	Current

The views can also be used to generate reports using EA's RTF report generator.

The maintenance cycle within Model Guardian keeps track of each version of the framework and updates the models along with their built-in and user-defined SQL views. The evolution of your framework and resulting models is trouble free.

Convenience Features

As mentioned above, the Object Types and Connector Types Tabs are coordinated with the Meta Type Hierarchy tree on the left so you always know just where you are. Each time you create a new framework element or update one, the changes appear in all of the lists that contain elements of its type, making them immediately available. You can click on underlined labels to see the Object Type or Connector Type in the hierarchy tree.

There are tooltips for all of the screen elements to provide instantaneous help. Detailed help is just click away.

In Conclusion

Enterprise Architect is one of, if not, the best UML design tool available. *Model Guardian* lets you harness its power to create modeling environments that will make your design teams' efforts more consistent, more productive, and more conforming to your standards. Perhaps even more importantly, it makes keeping your models up to date with the evolution of your frameworks a natural by-product of that evolution.

For more information, please visit www.OADConsulting.com, www.EA2.us, or contact OAD Consulting, Inc. at sales@OADConsulting.com.