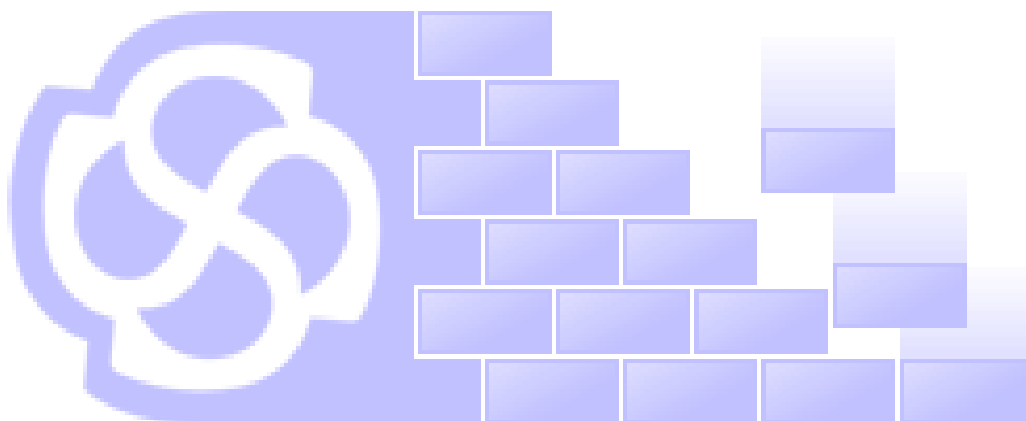# Enterprise Architecture Framework Design with Sparx Systems Enterprise Architect

by Sparx Systems
All material © Sparx Systems 2009
www.sparxsystems.com

Introduction

There exist numerous industry standard Enterprise Architecture Frameworks, such as Zachman Framework, DoDAF, MODAF and TOGAF. To support these predefined architecture frameworks, Sparx Systems provides specific plug-ins for the Enterprise Architect modeling platform.

However, many organizations need to create their own custom architecture frameworks, or variations to the standard frameworks, in order to meet specific organization requirements. Sparx Systems' Enterprise Architect provides powerful customization capabilities, which enable users to create architecture frameworks specific to their enterprise.

This article provides techniques for designing and implementing user-defined architectural frameworks using Enterprise Architect's extension capabilities. Knowledge of the Unified Modeling Language (UML), UML Profiles and some familiarity with Enterprise Architect is assumed.

A simple framework for demonstration

For the purpose of demonstration, consider the following simple architecture for defining requirements in an enterprise. Let's assume this simple framework aims to capture the dynamic nature of the requirements model of an enterprise. So we call it the *Dynamic Requirements Framework (DRF)*.

Dynamic Requirements Framework	Customer View	Internal Stakeholder View
Change	*Customer Change Model*	*Internal Change Model*
Feature	*Customer Request Model*	*Internal Request Model*

DRF only covers the requirements and change domain of an enterprise and therefore would not be considered a complete enterprise architecture framework. In practice, it is usual to create a broader framework that covers all aspects of the enterprise, as seen in Zachman Framework and TOGAF. These predefined frameworks are already supported by Sparx Systems Enterprise Architect through MDG Technology plug-ins.

Architecture of Dynamic Requirements Framework

The Dynamic Requirements Framework has two views, 'Customer' and 'Internal Stakeholder', which focus on the 'Change' and 'Feature' domains respectively. Therefore, we get four perspectives to address:

1. Customer triggered change
2. Customer triggered request for new feature
3. Internal stakeholder triggered change
4. Internal stakeholder triggered request for new feature

These perspectives translate to four structural blocks that make up the DRF framework:

1. Customer Change Model
2. Customer Request Model
3. Internal Change Model
4. Internal Request Model

Customer Change Model (CCM)

The CCM addresses customer triggered changes and their impact on the enterprise project. A customer triggered change usually impacts one or more existing requirements defined for the project. This impact will have associated costs in terms of resources, time and budget. An analysis is made before any decision is taken on this change. The DRF captures the analysis information and documents it under the Customer Change Model.

Customer Request Model (CRM)

The CRM addresses customer triggered requests for new features and their impact on the enterprise project. A customer request for a new feature usually results in a new requirement for the project, which has associated costs (resources, time, money). An analysis is made before approving the new feature request. The DRF captures the analysis information and documents it under the Customer Request Model.

Internal Change Model (ICM)

The ICM addresses the changes triggered by internal stakeholders and their effect on the enterprise project. Whenever a change is triggered by an internal stakeholder, the usual impact is on existing rules, policies or guidelines of the enterprise, which may incur costs. An analysis is made before any decision is taken on this change. The DRF captures the analysis information and documents it under the Internal Change Model.

Internal Request Model (IRM)

The IRM addresses requests triggered by internal stakeholders and their effect on the enterprise project. A request for a feature from an internal stakeholder usually results in the creation of a new project requirement. The request may be for a resource or for an activity, which will incur a cost. An analysis is made before approving the new feature request. The DRF captures the analysis information and documents it under the Internal Request Model.

Relationships between these four models should also be captured. For example, when a new requirement is created, an internal stakeholder may request appropriate resources.

Now that the plan for the basic structure and purpose of the framework is ready, our next objective is to construct a tool that provides the ability to model the Dynamic Requirements Framework. Enterprise Architect's Model Driven Generation (MDG) extensions enable us to customize the

modeling environment to support Dynamic Requirements Framework. Our aim is to create a set of toolboxes that group DRF model elements and respective diagrams.

Below is a preview of Enterprise Architect, customized to support DRF.
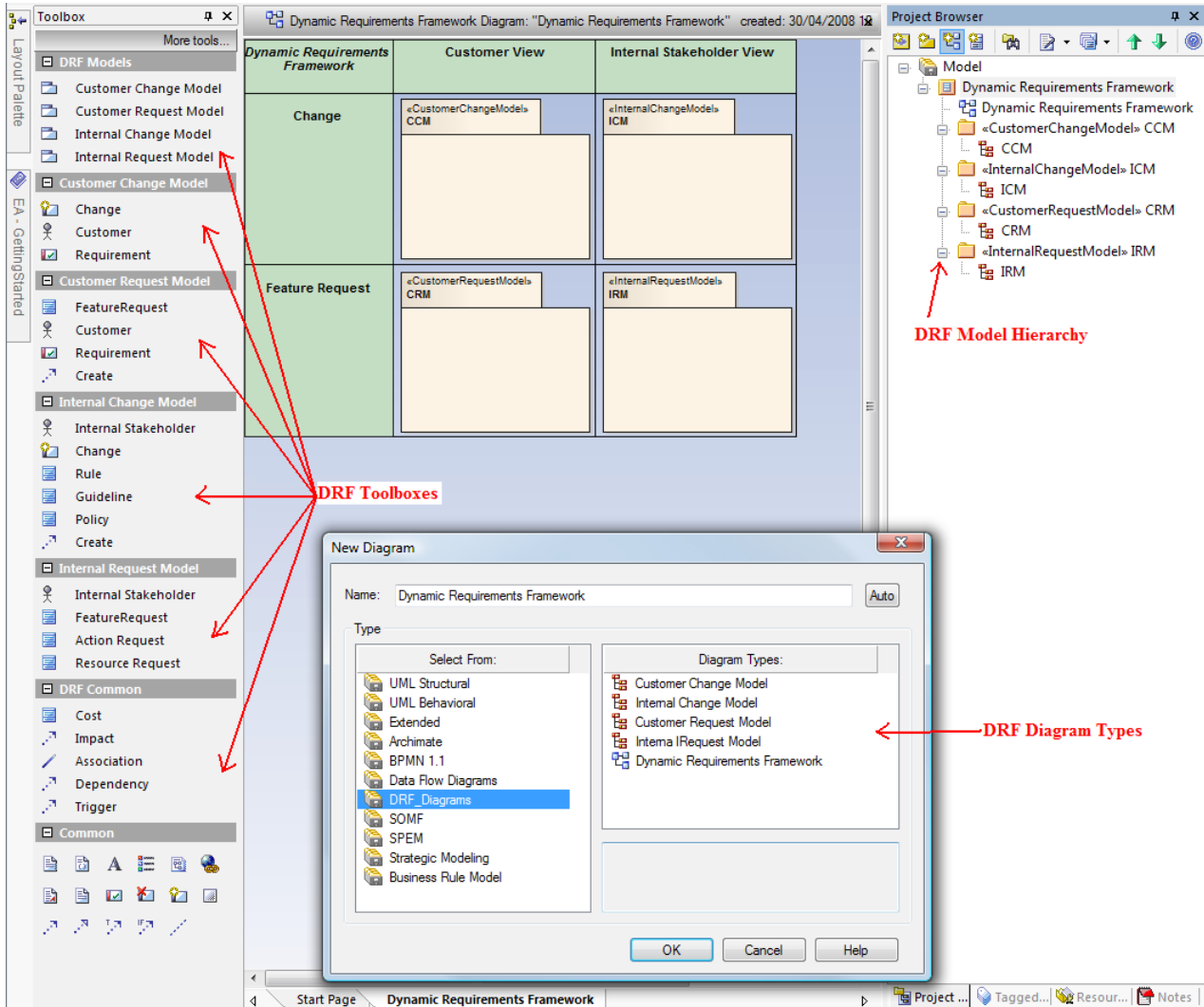
Extending Enterprise Architect

Model Driven Generation (MDG) Technologies provide a way to package together various modeling tool extensions, such as domain-specific UML profiles, new diagram types and alternate rendering of shapes – all of which are useful when defining a custom Enterprise Architecture. Enterprise Architect's automation interface can be used to further extend the modeling environment. This section discusses the methods to create and deploy the Dynamic Requirements Framework Technology.

The following basic steps are required to support the framework using Enterprise Architect extensions.

1. Develop the UML profiles
2. Create the MDG Technology
3. Deploy the MDG Technology
4. Create the Base Model (Template)

1. Develop the UML profiles

To develop a UML profile, we first need to identify the modeling constructs required by the framework. Let's analyze the four models in our framework to identify these constructs.

In the Customer Change Model, a *Customer triggers* a *Change* which will have an associated *Cost*. If the Change is approved then it *impacts* an existing *Requirement*.

The constructs we need to model the CCM are:

➢ Customer
➢ Change
➢ Cost
➢ Requirement

The CCM relationships are:

➢ Trigger
➢ Impact
➢ Association

Applying the same approach to the remaining models we get:

ICM constructs	CRM constructs	IRM constructs
➢ Internal Stakeholder ➢ Rule ➢ Policy ➢ Guideline ➢ Cost	➢ Customer ➢ Feature Request ➢ Requirement ➢ Cost	➢ Internal Stakeholder ➢ Feature Request ➢ Action Request ➢ Resource Request ➢ Cost
ICM relationships	**CRM relationships**	**IRM relationships**
➢ Trigger ➢ Impact ➢ Create ➢ Association	➢ Trigger ➢ Create ➢ Association	➢ Trigger ➢ Association

With the necessary model constructs identified, we can now build UML profiles for our framework. These profiles represent our identified constructs as stereotyped UML elements. Stereotypes can have customized shapes, and extended properties known as Tagged Values. For example, if we

need to uniquely identify all framework elements, we can create a Tagged Value named "ID" with type "String" for all elements in the profile.

The following diagram shows that the Customer and InternalStakeholder stereotypes extend the UML Actor metaclass. The attribute named ID with type string defines our identifier Tagged Value.
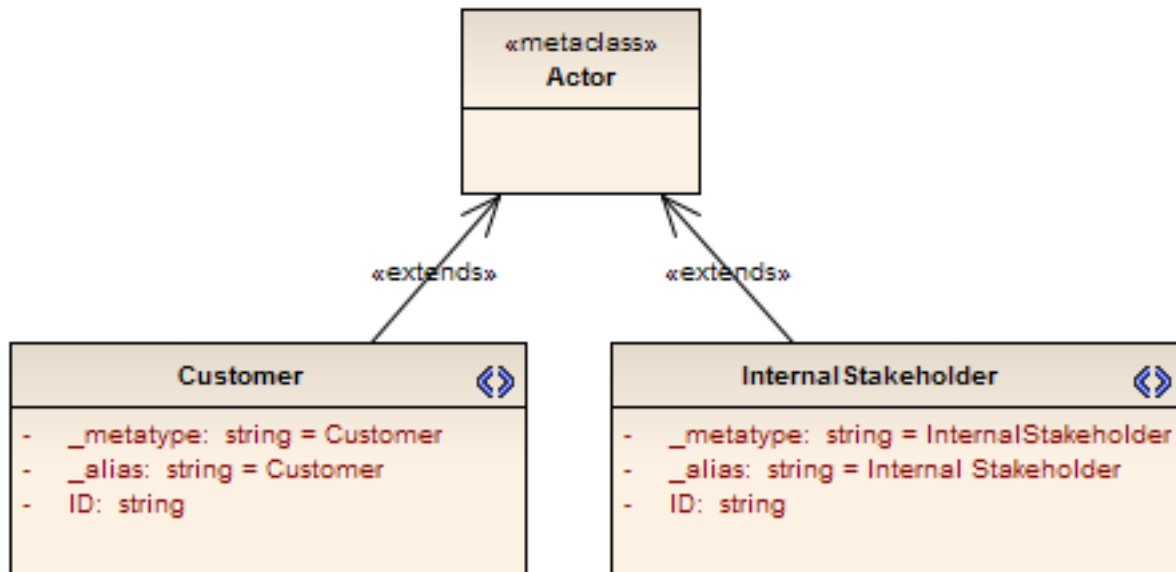
Using Enterprise Architect's supported attributes for stereotypes, it is possible to define image, icon, shape, size and color for the constructs.

More Information: Creating UML profiles in Enterprise Architect.

Enterprise Architect's Quick Linker provides a simple and fast way to create new elements and connectors on a diagram. This feature can be customized specifically for our framework, to guide the selection of valid elements and relationships. We would define quick links as part of a UML Profile and deploy it with the technology.

More Information: Defining quick links in Enterprise Architect.

Having created the stereotype elements, we can create new diagram types to visually model the framework. New diagram types are created by extending the existing Enterprise Architect diagrams, using a special UML profile for diagrams. For our Dynamic Requirements Framework, we create a new diagram type for each of the models:

- Customer Change Model Diagram
- Customer Request Model Diagram
- Internal Change Model Diagram
- Internal Request Model Diagram

We also need a diagram type, which we'll call "Dynamic Requirements Framework", to hold all

four models in a framework structure.

The following shows our DRF diagrams profile, as modeled in Enterprise Architect.
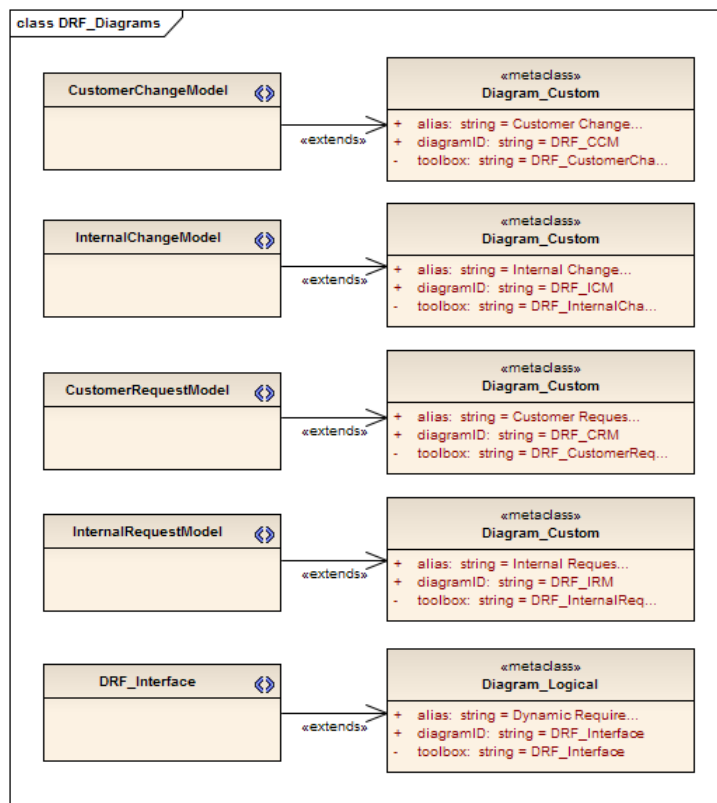
*More Information:* Creating new diagram types in Enterprise Architect.

The next step is to create a toolbox for each diagram type in the Dynamic Requirements Framework. We need toolbox pages for the each of the diagram types, to facilitate drag-and-drop functionality for the corresponding profile elements.

1. Customer Change Model toolbox page
2. Customer Request Model toolbox page
3. Internal Change Model toolbox page
4. Internal Request Model toolbox page
5. DRF Interface toolbox page

Each toolbox page is defined in Enterprise Architect by a profile, which references the applicable stereotypes. For the DRF models we have the below elements common:

➢ Cost
➢ Impact
➢ Association
➢ Trigger

These elements are defined in a common category and added to all DRF toolbox pages. The below example shows the toolbox profile for DRF's Internal Request Model, which references the DRF common elements.
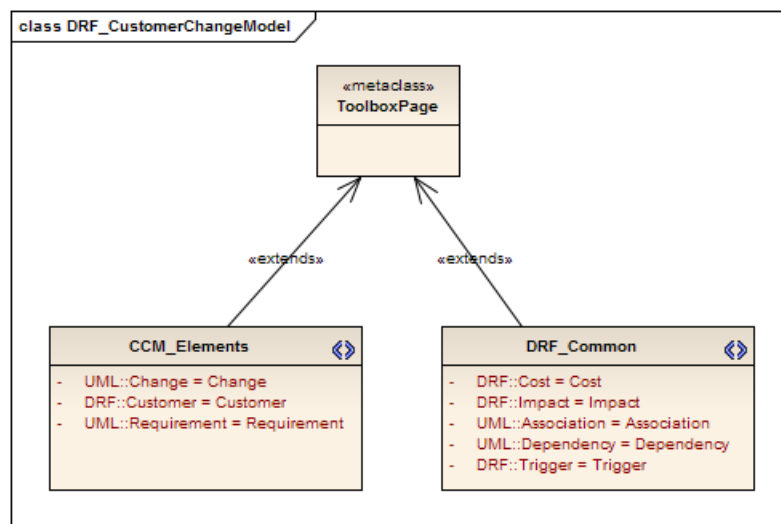
*More Information:* Creating UML profiles for toolboxes in Enterprise Architect.

Additionally, we can create a profile that defines Model Tasks, which are specific to the framework. These tasks then become available from Enterprise Architect's Tasks Pane.

More Information: Defining Model Tasks in Enterprise Architect.
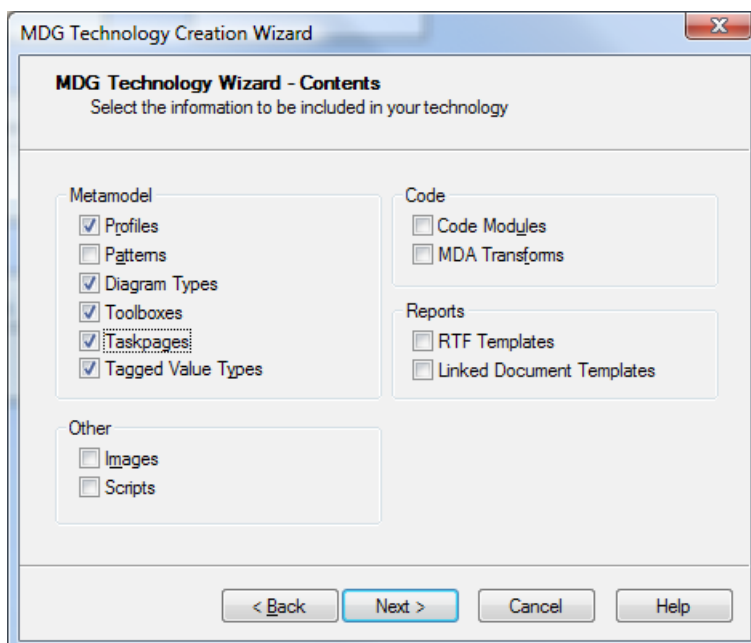

2. Create the MDG Technology

After creating UML profiles for stereotypes, diagram types and toolboxes, we are now ready to create the Dynamic Requirements Framework MDG Technology using Enterprise Architect's MDG Technology Wizard. We will use the wizard to create an MDG Technology file, which provides us with a lightweight 'plug-in' resource for Enterprise Architect capable of supporting the Dynamic Requirements Framework.

To invoke the wizard, select from the main menu: Tools | Generate MDG Technology File. On working through the MDG Technology Wizard, we will create DRF.MTS (MDG Technology Selection) file and add the UML profiles for Stereotypes, Diagrams, Toolboxes and Model Tasks.
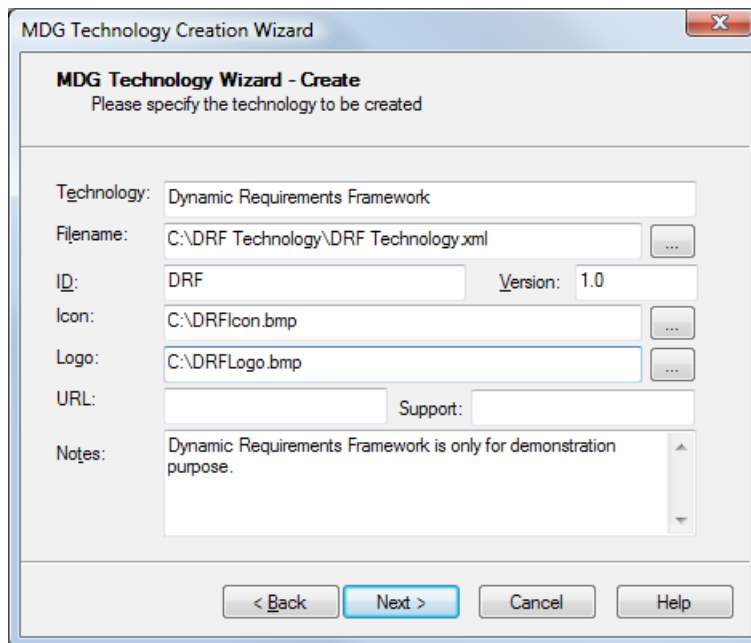
In addition to our profiles, the MDG Technology Wizard also allows us to add an icon, logo, patterns, images, scripts, tagged value types, RTF templates and Document templates.

The MDG Technology file created by the wizard is an XML file that enables Enterprise Architect to recognize the stereotypes, diagrams, toolboxes, model templates (discussed later in this article) and tasks for our framework.

More Information: Creating MDG Technology in Enterprise Architect.

3. Deploy the MDG Technology

An MDG Technology can be deployed either via a file or via an add-in. To deploy our DRF technology as a file, simply place the MDG Technology file created by the wizard in a folder named MDGTechnologies under the Enterprise Architect installation directory. By default, the Enterprise Architect installation directory is C:\Program Files\Sparx Systems\EA. When we restart Enterprise Architect, the DRF MDG Technology is available.

More Information: Deploying MDG Technology in Enterprise Architect.

Deploying the DRF technology via an add-in, allows more functionality to be included, such as:
  ➢ The ability to implement and configure model validation rules
  ➢ Implementation of technology-specific analysis tools

For example, we may have the following rules for DRF:
  ➢ "No new requirements must be created in CCM"
  ➢ "Every Change and Feature Request must have a Cost associated"
  ➢ "A change with rejected status must not impact a requirement"
  ➢ "A customer feature request with approved status must create a requirement"

These validation rules can be configured by an add-in.

More Information:
Defining Model Validation configuration in Enterprise Architect.
Implementation of Model validation.

## 4. Create the Base Model (Template)

It can be helpful to deploy with our MDG Technology a base model, which contains the stub framework structure. This base model serves as a template, allowing users to quickly instantiate our framework on a new project. To deploy a base model we need to include it with our DRF Technology. We start by simply constructing the framework using our profiles, as follows:

1. Create a new Enterprise Architect project.
2. Create a new view for the model and name it "Dynamic Requirements Framework".
3. Create the Dynamic Requirements Framework diagram from the Add Diagrams Wizard.
4. Right-click the diagram and select "Swimlanes and Matrix..."
   • Select the Matrix tab and enable the "Active" option.
     In the "Edit Selected Heading" group, select Heading for Type option.
   • Now edit title as "Dynamic Requirements Framework".
   • In the operations group, press the save button.
   • Then press "New" and edit rows and columns by selecting appropriate options for "Type" in "Edit Selected Heading" group.
   • Then press the "Save" button in Model Profiles option and name it "DRF"
   • After adjusting the row and column sizes, enable the "Lock" option in Matrix Options group in the "Swimlanes and Matrix..." dialog.
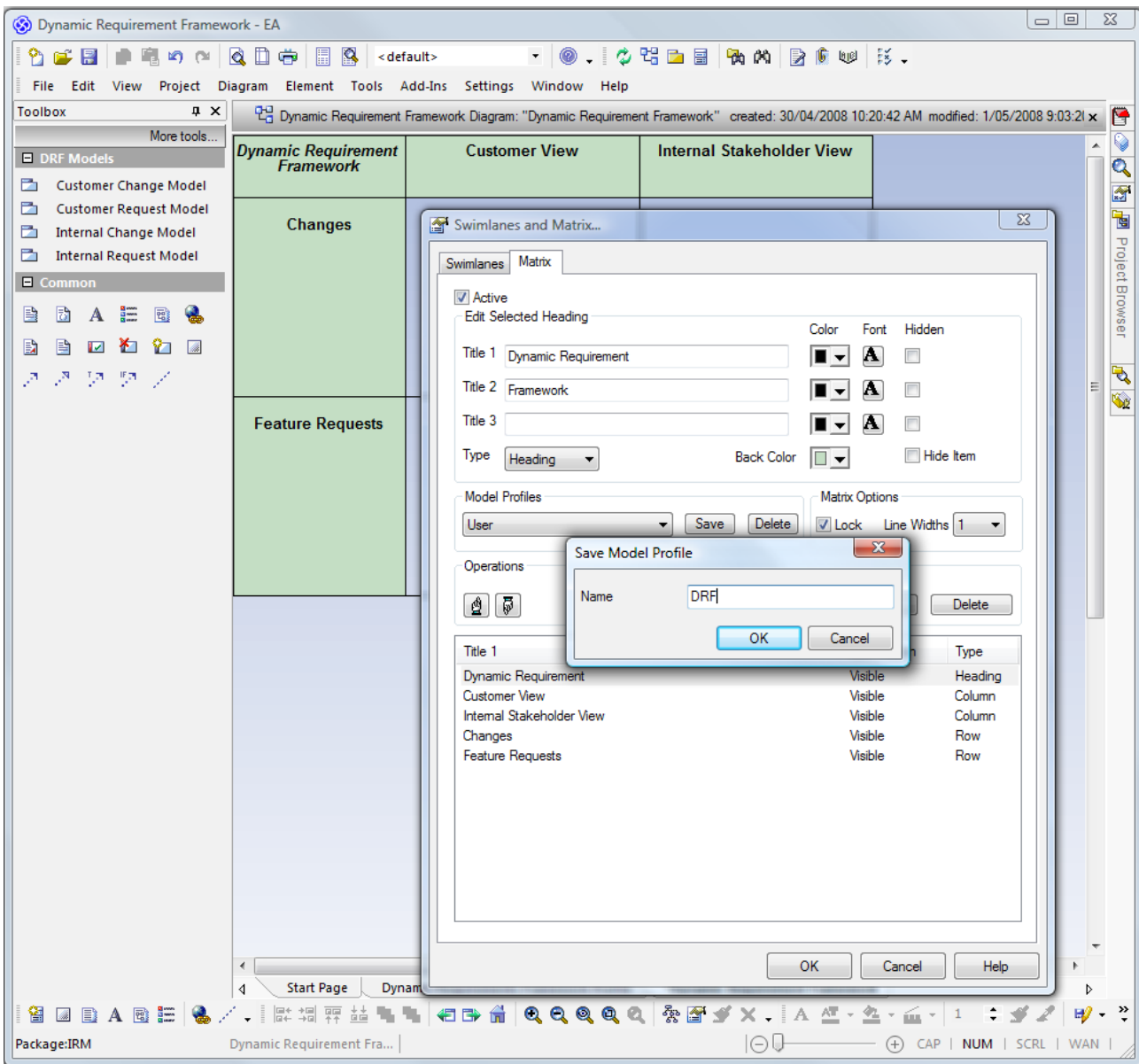
5. Now from the DRF toolbox create the model packages in the respective cells of the DRF matrix namely: "Customer Change Model", "Customer Request Model", "Internal Change Model" and "Internal Request Model".

Each of the DRF model packages has an associated default diagram type (see the companion profile implementation with this document for details). Therefore, whenever a DRF model package is created, the appropriate default diagram will automatically be created under it. The resulting DRF model hierarchy is shown in Enterprise Architect's Project Browser below.
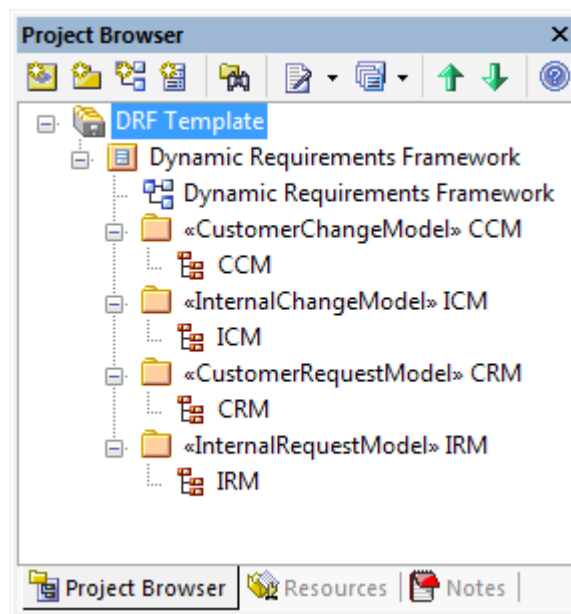

Below is a snapshot of the DRF interface diagram in Enterprise Architect.
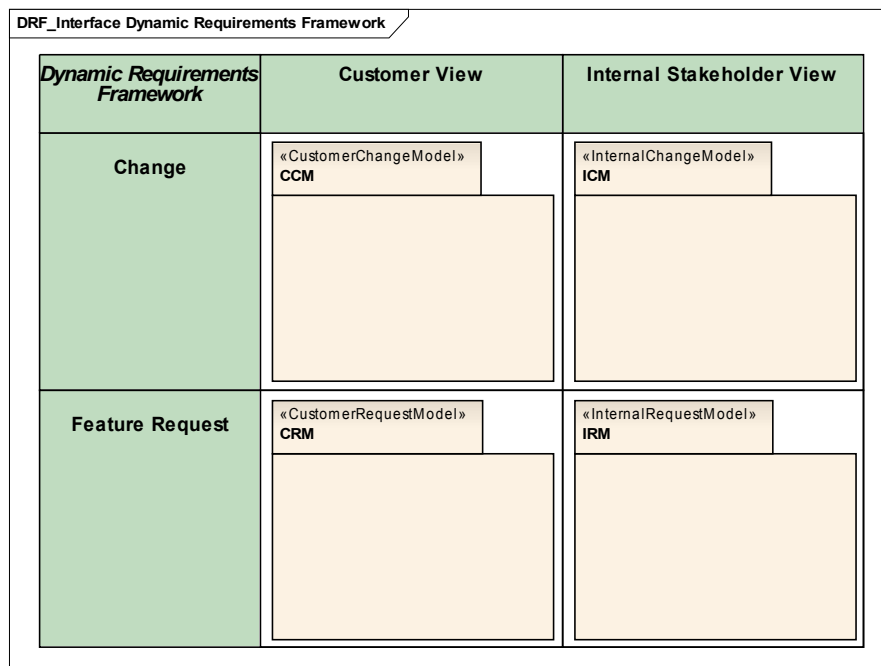
Note: Enterprise Architect diagram elements have an option, "Selectable", which prevents them from being selected and moved. Turning this option 'on' for elements in the interface diagram will help to prevent accidental modification.

If desired, alternate images can be assigned to packages in the interface diagram. Below are screen shots of the Zachman Framework and TOGAF interface diagrams with alternate images. Note: A framework interface need not use swimlanes or matrices, as seen in the TOGAF interface diagram.
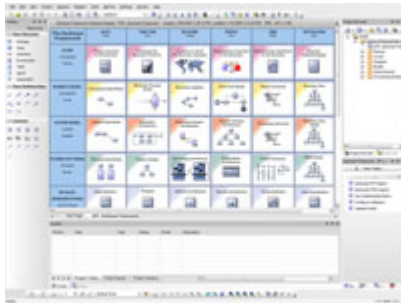
The Zachman Framework
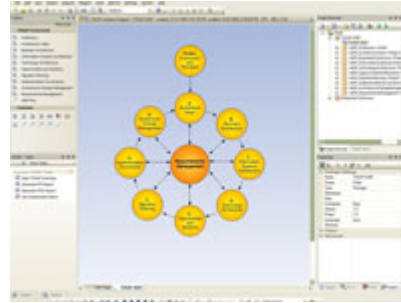
TOGAF

Now we have a framework model in place for reuse. To reuse the model structure as a template, we first need to export it as an XMI file:

1. Right-click on the model's root node and select "Export Model to XMI"
2. Supply a filename such as "DRF_ModelTemplate.xml"

By referencing this XMI file in our DRF technology, it will be available in any Enterprise Architect project that has the DRF Technology enabled. Enterprise Architect can handle any number of model templates for a given MDG Technology.

To make the DRF technology aware of the model template, we have to add the template to the MDG Technology Selections. Open the *DRF.mts* file in a text editor and copy the following <ModelTemplates> block inside the <MDG.Selections> block:

<ModelTemplates>
   <Model name="Dynamic Requirements Framework model"
        description="Demonstration"
        location="C:\DRF Technology\DRF_ModelTemplate.xml"
        default="yes"
        icon = "34"
        filter= "Common"/>
  </ModelTemplates>

Now update the MDG Technology file for DRF using the MDG Technology Wizard. Update the file in the MDG technologies folder under the Enterprise Architect installation directory as in section 3 and restart Enterprise Architect for the update to take effect.

More Information: Working with MTS files in Enterprise Architect.

Having deployed the MDG Technology file with DRF-specific profiles, corresponding toolboxes and diagram types, and a base model template, the Dynamic Requirements Framework is now ready for use!

The example DRF model used throughout this document, along with the associated resources that implement the MDG Technology, is available for download from:
http://www.sparxsystems.com/downloads/whitepapers/assets/ea_framework_sample.zip

The zip file from the above URL includes the following:

- **Definition of UML profiles for DRF**: See the file "Dynamic Requirements Framework.eap" which contains the node "Dynamic Requirements Framework Profile"

- **Model Template for DRF**: See "Dynamic Requirements Framework.eap" under the node "DRF Template" The exported XMI file is named: "DRF_ModelTemplate.xml"

- **MDG Technology for DRF**: See the file "DRF Technology.xml"

- **MDG Technology Selections**: See the file: "DRF.mts"