Modeling an ArcGIS™ geodatabase with UML® and Enterprise Architect

Sparx Systems MDG Technology® for ArcGIS supports visual modeling of geodatabases in Enterprise Architect. It provides a UML profile for modeling ArcGIS concepts, the ability to generate ArcGIS schemas as XML Workspace documents, and reverse engineering of legacy geodatabases into a visual UML model.

This tutorial explains how to model an ArcGIS geodatabase in Enterprise Architect and generate a corresponding ArcGIS schema. It also explains how ArcGIS concepts relate to UML notation in Enterprise Architect.

Examples in this tutorial were created using Enterprise Architect 9.3.934

Table of Contents

Acknowledgements

Support for modeling ArcGIS databases in Enterprise Architect was developed in collaboration with the Commonwealth Scientific and Industrial Research Organisation (CSIRO), who defined mappings between UML 2 and ArcGIS concepts, and prototyped an automated import and export capability for ArcGIS geodatabase schemas represented in UML.

Matt Langley and *Anthony Burgon* of *Geoplex*, and *Frank Truyen* of *Cephas Consulting Corp* helped to improve early implementations of MDG Technology for ArcGIS via reviews and suggested enhancements to the ArcGIS profile and related functionality.

As subject matter experts, the following people have provided valuable suggestions to expand Enterprise Architect's UML Profile for ArcGIS and to improve the effectiveness of model-driven generation of geodatabase schemas for ArcGIS:

- *Khin Fah Bong* of *Shell.*
- *Narmina Lovely* of *BP Exploration & Production Inc.*
- Staff at *Willbros Group, Inc.*

Tobias Spears, of *Fisheries and Oceans Canada* contributed valuable feedback on achieving model reuse by ensuring compatibility of metadata across modeling tools.

Introduction

Enterprise Architect 9.3 provides built-in support for modeling ArcGIS geodatabase designs. Using a UML profile for ArcGIS, Enterprise Architect helps you to:

- Create a visual UML model of your ArcGIS geodatabase
- Export your modeled ArcGIS schema as an XML Workspace document that is readable by ArcGIS 10.0 and ArcGIS 10.1.
- Import your existing ArcGIS database schemas to an Enterprise Architect model.

Visualizing your geodatabase designs with Enterprise Architect helps you to:

- Facilitate traceability of your GIS systems to the broader enterprise model
- Communicate geospatial design concepts to a wide audience of stakeholders
- Leverage model driven architecture to improve development and maintenance of geodatabases.

The purpose of this tutorial is to help you learn how to model geodatabases in Enterprise Architect that are suitable for use with the Esri® ArcGIS suite of tools. The tutorial also provides details of how ArcGIS concepts relate to UML and Enterprise Architect features.

What you will learn

We will use a very simple example, based on a Smart Meter, to demonstrate how to model an ArcGIS database in Enterprise Architect.

This tutorial will teach you how to:

- Create a model that represents your ArcGIS Workspace
- Define a visual model of your features, tables, indexes and domains
- Configure spatial reference information in your model
- Export the UML model to an ArcGIS schema, using the ArcGIS XML Workspace document.

What you need

- Enterprise Architect **9.3**, Build 934 or later
- ArcGIS **10.0** or later (to import generated schemas).

Step 1. Create an ArcGIS Workspace model

Enterprise Architect's Model Pattern Wizard helps you to create a package structure that supports geodatabase design for ArcGIS. We begin by using the Model Pattern Wizard to create a starter model for our geodatabase:

1. Open Enterprise Architect.
2. Click on **New Project**.
3. Specify a file name, such as *My ArcGIS Project*, and click **Save**.
4. You are prompted by the **Model Wizard** to add a model pattern.
5. Choose *ArcGIS* from the Technology window and select *ArcGIS Workspace* from the Name window. Click **OK**.
6. Enterprise Architect's **Project Browser** is populated with a top-level package named *ArcGIS Workspace*. The package is stereotyped as *ArcGIS*, shown within guillemets, « ».*

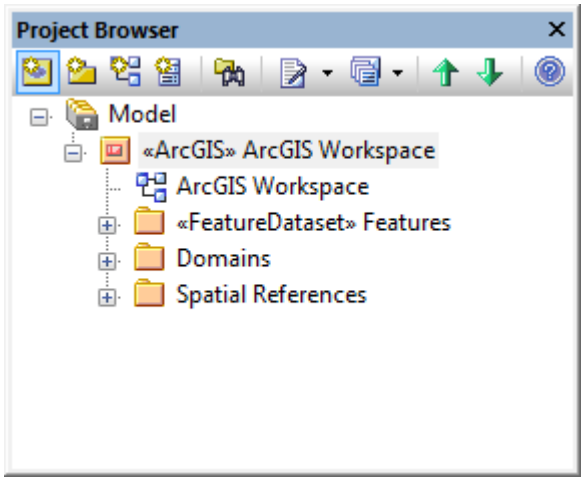

Figure 1: UML Package structure created for an ArcGIS model.

7. You may rename this package, however its stereotype must remain as “ArcGIS”.

   Note: Expanding the ArcGIS Workspace package reveals the sub-packages, *Features, Domains* and *Spatial References*. These sub-packages will contain elements that define specific characteristics of your schema. We will discuss their contents in subsequent sections.

   You have considerable flexibility in how you structure sub-packages of the ArcGIS model. The model pattern provides a suggested set of sub-packages for organizing your schema model, however these can be renamed, moved or deleted if desired.
8. In the **Project Browser**, double-click the diagram node named *ArcGIS Workspace*.
9. Enterprise Architect opens a Package diagram for our Workspace model, shown in Figure 2.

* The stereotype is Enterprise Architect's means of relating this part of the model to ArcGIS.

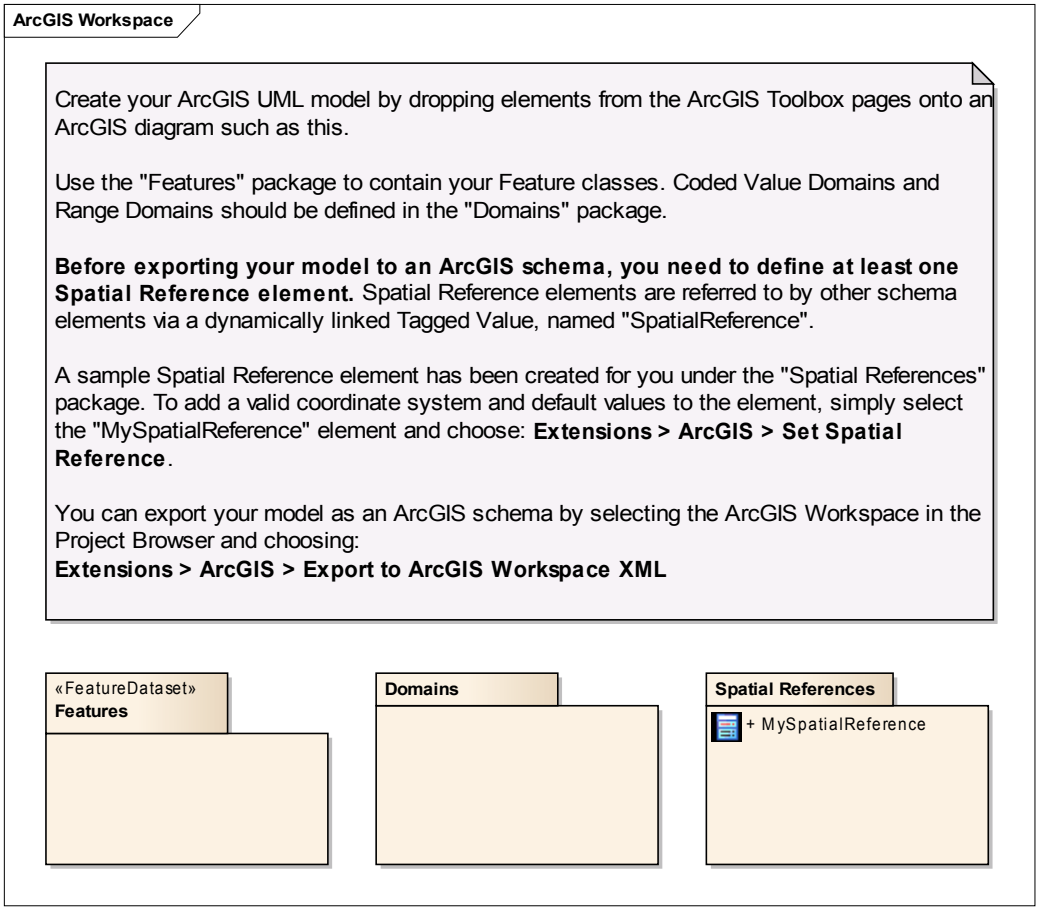

Figure 2: UML Package diagram created by the ArcGIS model pattern.

We have used the pattern to create a placeholder in which to model our ArcGIS geodatabase. Each of the sub-packages from the pattern contain stub diagrams that will guide us in building a valid ArcGIS model. From such a model, Enterprise Architect can generate a corresponding schema that is readable by ArcCatalog.

Step 2. Define a visual model of your features and tables

Create a Feature Class

We now use the structure created by the Model Pattern Wizard to complete our geodatabase design, starting with a Feature Class.

1. On the *ArcGIS Workspace* diagram, double-click the package named *Features* to open its child diagram. The ArcGIS toolbox should display when you open an ArcGIS diagram. (If the toolbox is not visible, display it using the menu **View > Diagram Toolbox**, or press **Alt+5**)
2. From the ArcGIS toolbox, drag and drop a Point element onto the *Features* diagram.

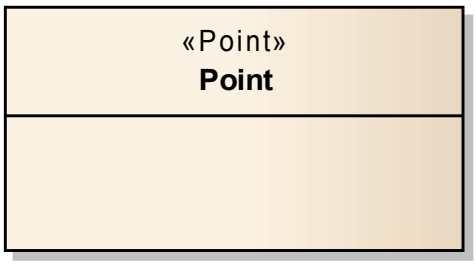

Figure 3: The ArcGIS Point Feature Type represented as a UML Class.

Define properties of the Feature Class

1. **Edit the properties of the Point element:** When you drop the Point element onto the diagram, a Properties window displays, depending on your model editing options. If not, double-click the element to display its properties.

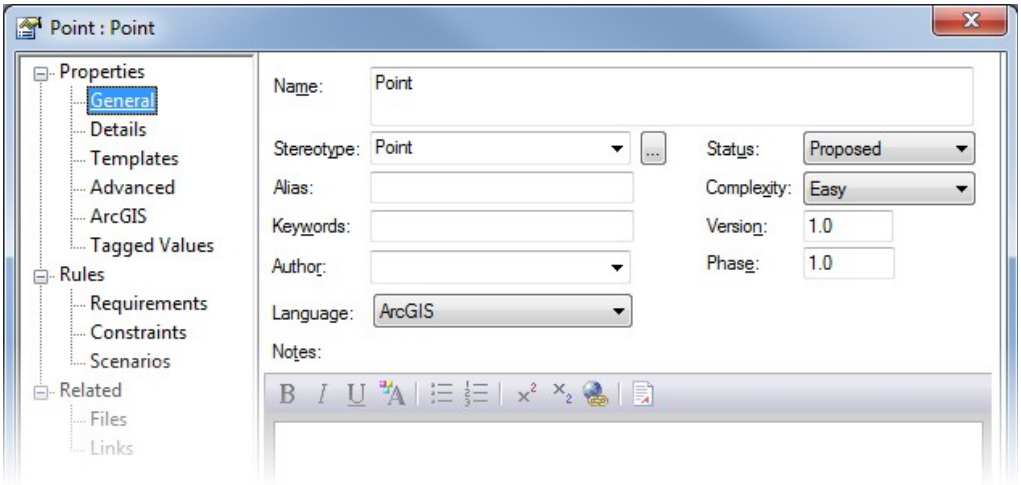

Figure 4: Model properties for Point Feature Type created in Enterprise Architect.

2 Rename the Point element by typing *House* into the **Name** field.

3 Notice the tab named **ArcGIS** on the left-hand panel. You can view ArcGIS-specific properties by clicking this tab. For this example, we need not adjust any of these properties.

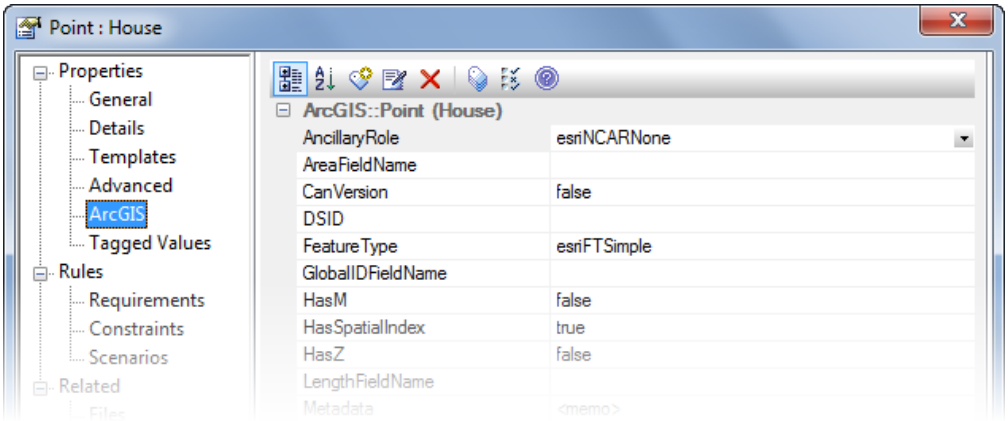

Figure 5: ArcGIS elements have extension properties, which are modeled as UML Tagged Values.

4 To save the changes and return to the diagram, click **OK** on the Properties window.

For the purpose of this example, our simple *House* Feature Class does not contain any custom fields. You can add custom fields to your Feature Classes by using the *Field* stereotype from the ArcGIS toolbox. This creates a UML Attribute with stereotype *Field* and ArcGIS-specific Tagged Values. We will create an example of a custom field in the next section, as we model an ArcGIS table for our geodatabase.

Notes:

- A summary of how each ArcGIS concept is captured in UML is provided in Appendix A.
- You can create custom indexes and link these to your Feature Class. For more information, see Appendix B.
- Modeling coded value domains and range domains is discussed in Appendix C.
- Modeling subtypes is discussed in Appendix D.
- Abstract classes can be used to simplify the model, where multiple Feature Classes relate to the same basic concept and have fields in common. This is discussed in Appendix E.
- You can customize and control the display of various 'system-level' fields, such as OBJECTID, as described in Appendix F.

Create a Table (Object class)

We want to relate our *House* Feature Class to a *Smart Meter* table in our ArcGIS database. To do this, we first create the element that represents the table and define its properties. The process is similar to creating a Feature Class, though we'll start with a new diagram:

1. **Create a new ArcGIS diagram:** Select the view named *ArcGIS Workspace* in the **Project Browser**. Click on the 'New Diagram' icon from the **Project Browser's** toolbox. You will be prompted to enter a name and select from one of the diagram types. Enter a name, such as *House – Meter,* choose the *ArcGIS* diagram type and click **OK**.
2. From the ArcGIS toolbox, drag and drop a *Table (Object Class)* element. In Enterprise Architect, an ArcGIS Table or Object Class, is a UML class with stereotype *ObjectClass.*

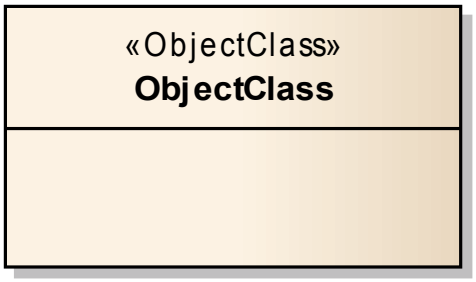

Figure 6: The ArcGIS Table represented as a UML Class.

Tip: A Table cannot exist inside a Feature Dataset in the ArcGIS schema. Therefore, when you create a new Table element in Enterprise Architect, ensure it is not contained within a Feature Dataset package. You can either use the Workspace package, or another unstereotyped package that is used for organizing the model hierarchy. You can change the package in which a Table resides simply by dragging it within Enterprise Architect's **Project Browser**.

Define properties of the Object class

1. Open the properties of the *ObjectClass* element.
2. Rename the element by typing *SmartMeter* into the **Name** field.
3. To save the changes and return to the diagram, click **OK** on the Properties window.

Create a Field for the SmartMeter Table

We will define an ArcGIS field in the table that represents the house to which a given smart meter belongs:

1. From the ArcGIS toolbox, drag and drop a *Field* onto the *SmartMeter* element. You will be prompted to enter a Name and Type.
2. Type *HOUSEID* into the **Name** field.
3. Use the drop-list on the **Type** field to select the type *esriFieldTypeInteger*. Click **OK**.

   Note: You can adjust the name and type information of fields at any time by selecting the Table element, then either right-click and choose **Attributes** from the context menu, or press F9.

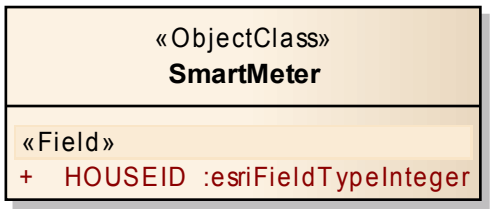

Figure 7: Fields are represented as stereotyped UML attributes.

Next, we model the relationship between our Table and Feature Class.

Model the 'House-to-SmartMeter' relationship

A stereotyped UML Association represents relationships between ArcGIS elements in Enterprise Architect. We will create a relationship between the *House* and *SmartMeter* elements to reflect the fact that each house can have zero or more smart meters.

1. **Include the *House* element in the current diagram:** Locate the *House* element in the **Project Browser**. Hold the **Control** key and drag the element onto the diagram. When prompted, choose the option *as Simple Link* and click **OK**.

   (You may wish to suppress display of the system fields in this diagram link to the House class by using the menu: **Extensions > ArcGIS > Show or Hide ArcGIS System Fields**)

2. Drag the **QuickLinker** arrow **from** the *House* element **to** the *SmartMeter* element. As the Source element, *House* corresponds to the OriginClass in the geodatabase schema.

Tip: The order is important when creating RelationshipClass connectors. The class you start dragging from becomes the Source element – in ArcGIS terms that corresponds to the OriginClass. The element at the other end of the connector is considered the destination.

If you make a mistake as to which element you start dragging from when creating the connector, the fix is simple: Right-click the connector and choose **Advanced > Reverse Direction** from the context menu. The Source and Target roles – that is, Origin and Destination classes – are instantly reversed. If you have already set any related properties, such as origin and foreign keys, you may need to reset these as well.

3. Choose RelationshipClass from the menu when prompted. A connector will be created.

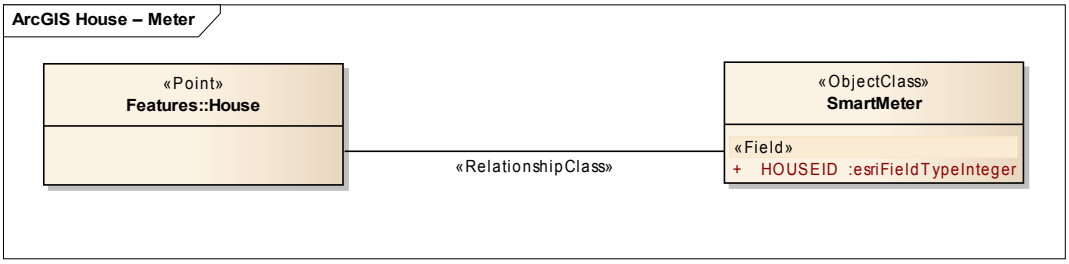

Figure 8: An ArcGIS Relationship Class is represented by a stereotyped UML Association.

4. **Edit the connector properties:** Depending on your model editing options, the Properties window displays automatically upon creating a connector. Otherwise, double-click the connector to edit its properties.

5. Name the relationship by typing *HouseToMeter* into the **Name** field.

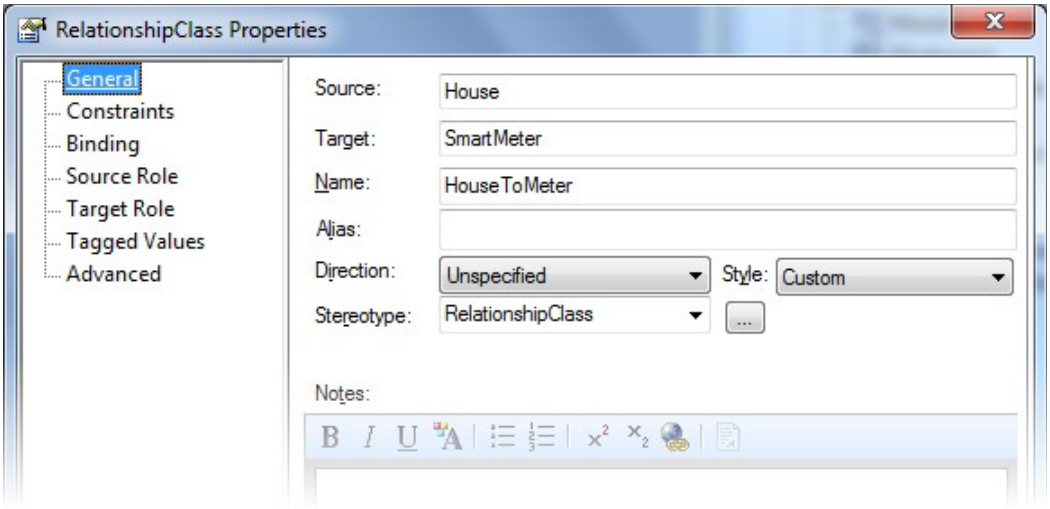

Figure 9: Connector properties let you to name the relationship, set multiplicity and define ArcGIS-specific properties.

Tip: The connector's **Direction** field corresponds to the ArcGIS Notification property of a RelationhipClass. By default, the Direction is *Unspecified*, corresponding to a value of esriRelNotificationNone. This value can be overidden using the "Notification" Tagged Value.

6. **Define the Source Role:** Click on the **Source Role** tab. Name the Role by typing *BelongsTo* into the **House Role** field. (This value corresponds to the ForwardPathLabel in the generated ArcGIS schema.)

   Set the **Multiplicity** to *1*, indicating that a smart meter belongs to only one house.

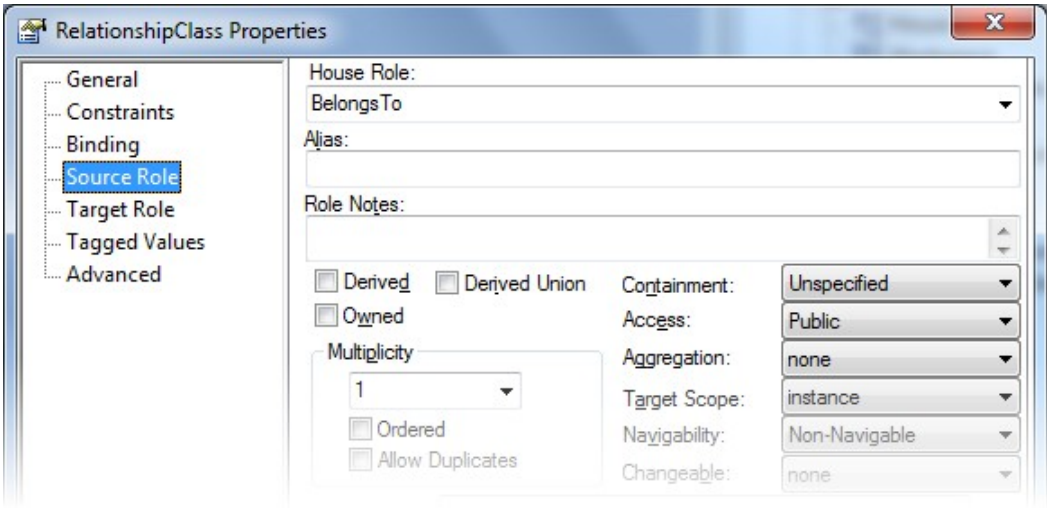

Figure 10: Use the Source and Target Role tabs to define details at each end of the relationship.

7. **Define the Target Role:** Click on the **Target Role** tab. Name the role by typing *HasMeter* into the **SmartMeter Role** field. (This value corresponds to the BackwardPathLabel in the generated ArcGIS schema.)

   Set the **Multiplicity** to 0..*, which indicates that each house has zero or more smart meters.

8. **Define other ArcGIS specific properties:** Click on the **Tagged Values** tab.

   1. **Define Origin Primary Key:** Click the ellipsis button (. . .) in the *OriginPrimaryKey* Tagged Value. Navigate to the *Features* package and expand the *House* class. Choose the *OBJECTID* attribute.

   2. **Define Origin Foreign Key:** Click the ellipsis button in the *OriginForeignKey* Tagged Value. Navigate to the *SmartMeter* class and expand it. Choose the *HOUSEID* attribute.

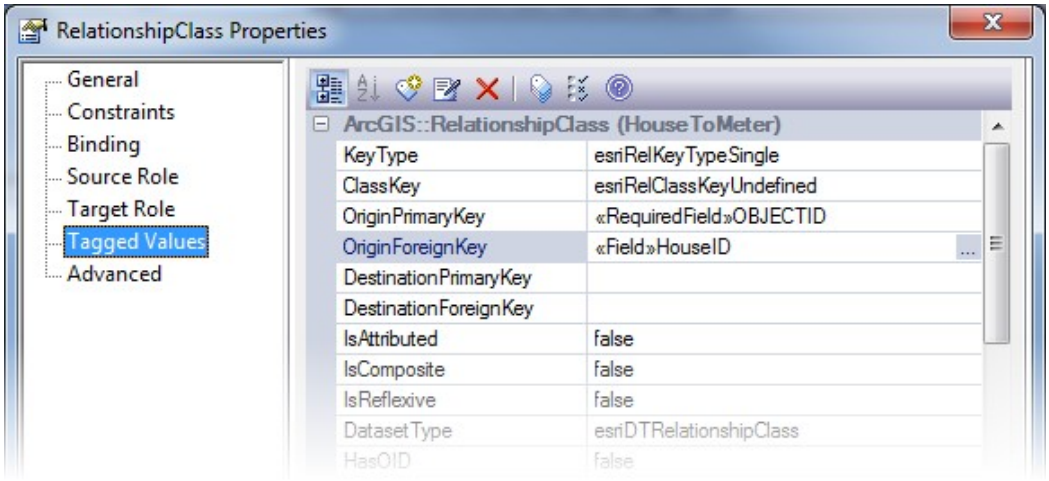

Figure 11: Tagged values on the connector contain the remaining ArcGIS-specific properties.

Tip: If you need to model a many-to-many relationship, use the alternative *Relationship Class* connector which is modeled using a UML *Association Class.* You must set the *OIDFieldName* value to the primary key defined by the Association Class, and the *DestinationPrimaryKey* and *DestinationForeignKey* values respectively.

Tip: You can use the *CatalogPath* tagged value to determine under which Feature Dataset this RelationshipClass will reside in the geodatabase. By default: 1-many relationships will reside under the Feature Dataset (UML package) in which the OriginClass (Source UML Class) is defined; for many-to-many relationships the UML Package containing the UML AssocationClass element defines the CatalogPath.

You can override these default paths using the *CatalogPath* tagged value, by choosing any Feature Dataset package in the Workspace. For example, you may create a dedicated Feature Dataset for containing RelationshipClasses. In a large system, with many RelationshipClasses , this may improve organization and readability of the implemented geodatabase schema in ArcCatalog.

9. Save the connector properties by clicking **OK.**

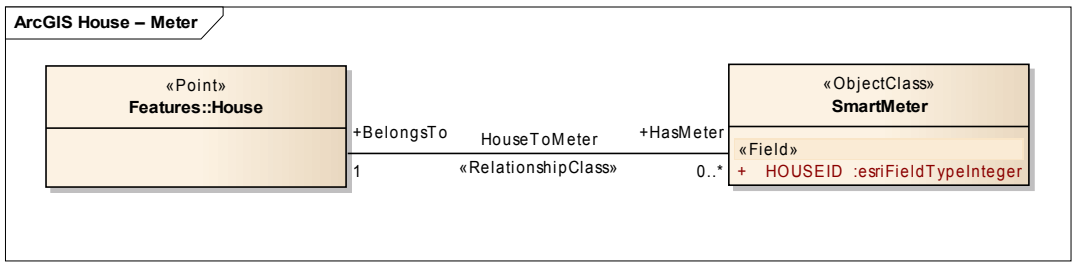

Figure 12: The completed Relationship Class between House and SmartMeter (Tagged Values are not displayed).

The model hierarchy that contains all the of the required elements to complete our Smart Meter example is shown in Figure 13. Notice that the SmartMeter Table is defined as a child of the Workspace package, not the Feature Dataset package.

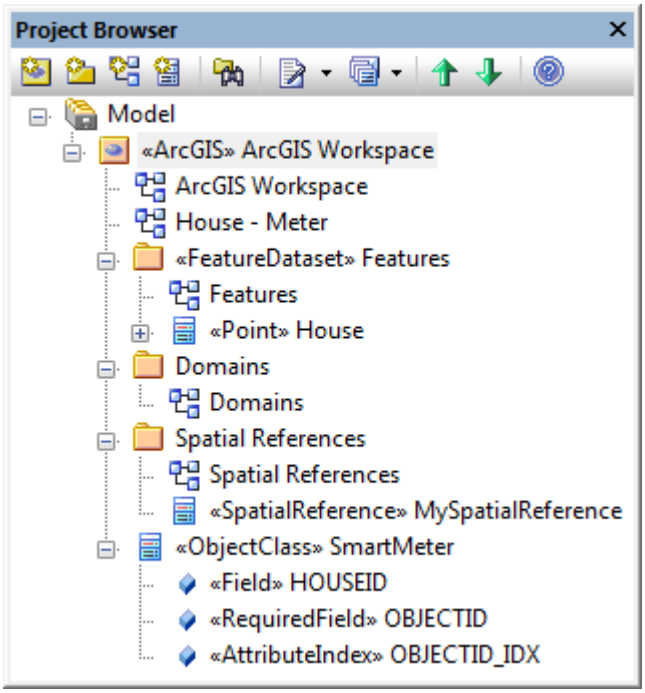

Figure 13: Hierarchy of model elements in the Smart Meter example.

All that remains now, is to configure spatial reference information and export the ArcGIS workspace.

Specify a Spatial Reference

To export a valid ArcGIS schema from Enterprise Architect, you must define at least one Spatial Reference element and refer to it in your schema model. We learn how to define the details of a Spatial Reference element in the next section. For now, we will simply assign to our Features package the Spatial Reference element created by the ArcGIS model pattern. This assignment applies to all Feature Classes contained in the Dataset.

1. **Edit the properties of the Feature Dataset package:** From the *ArcGIS Workspace* diagram, or the **Project Browser**, right-click the *Features* package. Select **Properties** from the context menu.
2. **Assign a Spatial Reference element:** Open the ArcGIS properties tab and click the ellipsis button in the *SpatialReference* Tagged Value. You will be prompted to select an element, of stereotype *SpatialReference,* from the model hierarchy.
3. Navigate to the *Spatial Reference* package. Select the element *MySpatialReference*. Click **OK**.

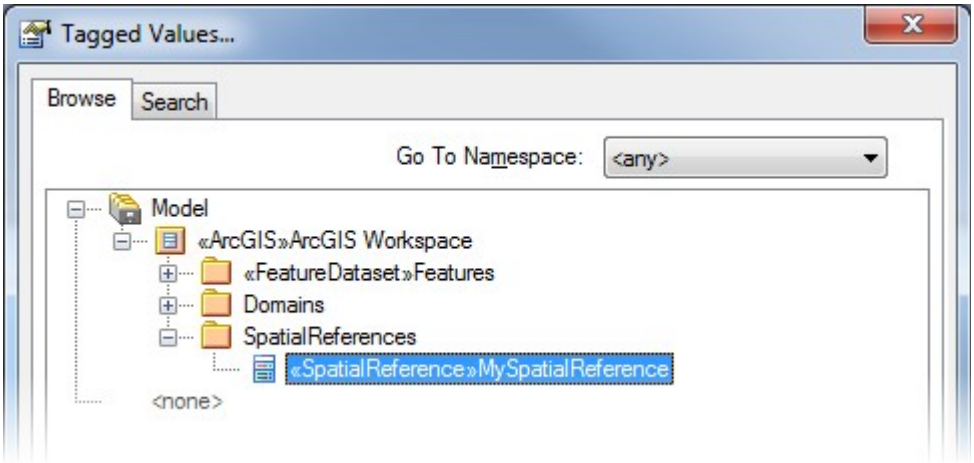

Figure 14: Your ArcGIS model must contain at least one SpatialReference element.

Step 3. Configure the SpatialReference element

Enterprise Architect's ArcGIS Workspace model can include one or more SpatialReference elements. A SpatialReference element specifies which coordinate system is used by elements in the Workspace and the values associated with that coordinate system, such as XY resolution and tolerances. When you create an ArcGIS Workspace using a Model Pattern in Enterprise Architect, a sample SpatialReference element is created for you. In the previous section, we linked this element to our Feature Dataset package. We will now configure the SpatialReference element to specify a coordinate system.

1. Open the diagram named *ArcGIS Workspace.*
2. *Open the Spatial References diagram by double-clicking the Spatial References package in the ArcGIS Workspace diagram.*

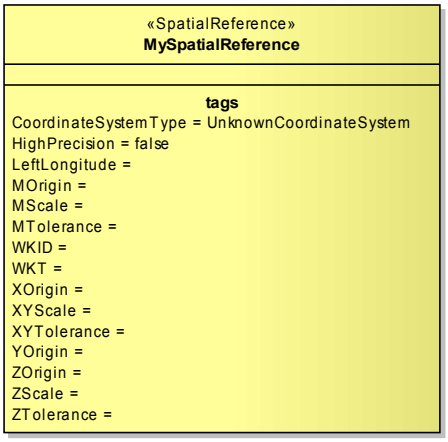

Figure 15: A SpatialReference is modeled as a UML Class and captures the coordinate system.

3. Select the element named MySpatialReference.
4. From the main menu select: **Extensions > ArcGIS > Set Coordinate System.**

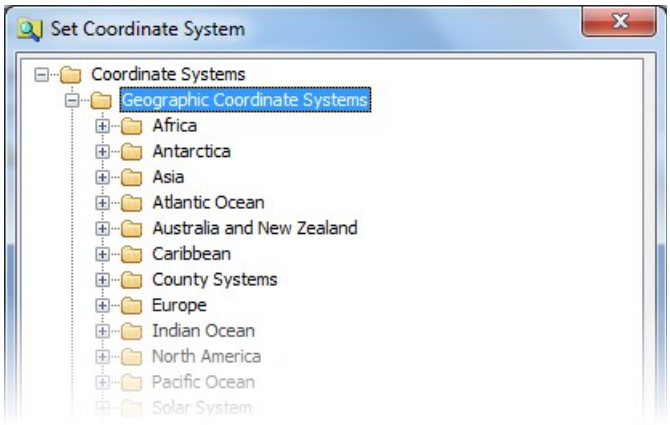

Figure 16: User interface for specifying a predefined coordinate system.

5. Use the tree view to select an appropriate coordinate system and click **OK**.
6. Notice that the SpatialReference element on the diagram now reflects the default values associated with that coordinate system. The well-known-text (WKT) along with other values are stored as UML Tagged Values on the SpatialReference element.

   The example below uses the WGS 1984 Geographic Coordinate System. Our Feature Dataset package now uses the WGS 1984 coordinate system because we previously linked it to *MySpatialReference* via its SpatialReference Tagged Value. This can be modified at any time simply by selecting another coordinate system using steps 1-5 above.

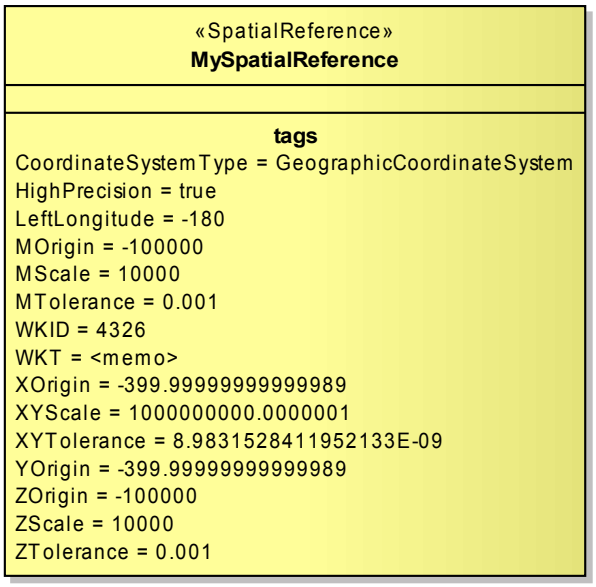

Figure 17: Default values for the WGS 1984 Geographic Coordinate System.

Step 4. Export to an ArcGIS XML Workspace document

Now that our feature model is complete, Enterprise Architect can generate the corresponding ArcGIS schema as an XML Workspace document:

1. Ensure that all open diagrams are saved. Simply right-click one of the open diagram tabs which are located at the bottom of the main view and choose **Save All.**
2. Select the top-level *ArcGIS Workspace* package in the **Project Browser**.
3. Invoke the ArcGIS exporter via the menu: **Extensions > ArcGIS > Export to ArcGIS Workspace XML**. In the list of XML Types, *ArcGIS* will be selected.

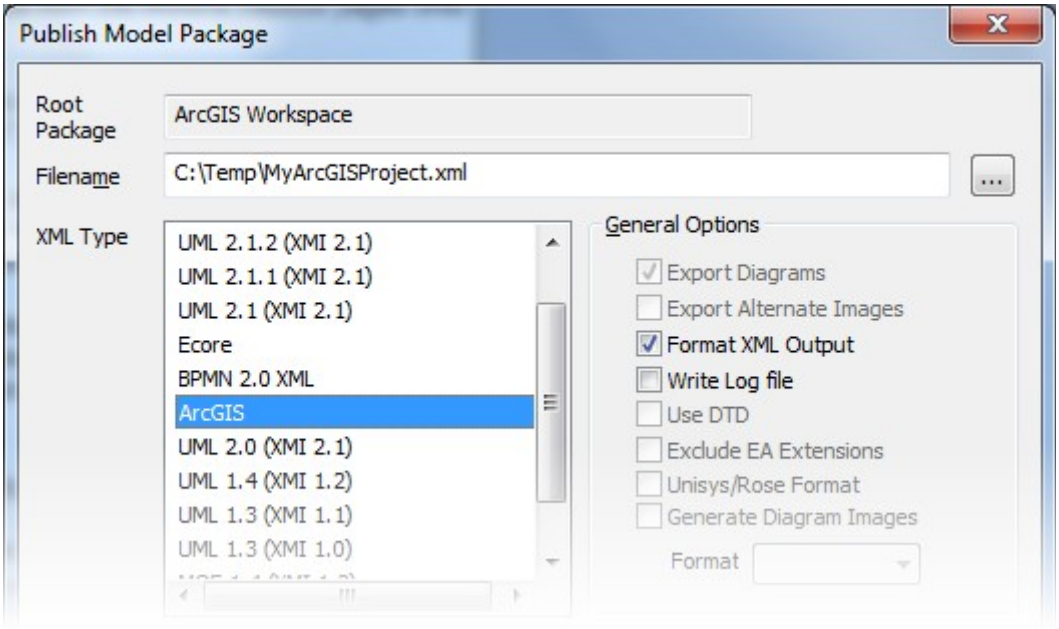

Figure 18: Export to an ArcGIS XML Workspace is supported by Enterprise Architect's Model Publisher.

4. Define an appropriate file path for the Workspace document and click **Export.**

Step 5. Import the XML Workspace document to ArcCatalog

1. Open **ArcCatalog** and create a new geodatabase.
2. Right-click the geodatabase in the **Catalog Tree**. **Choose Import > XML Workspace Document**.

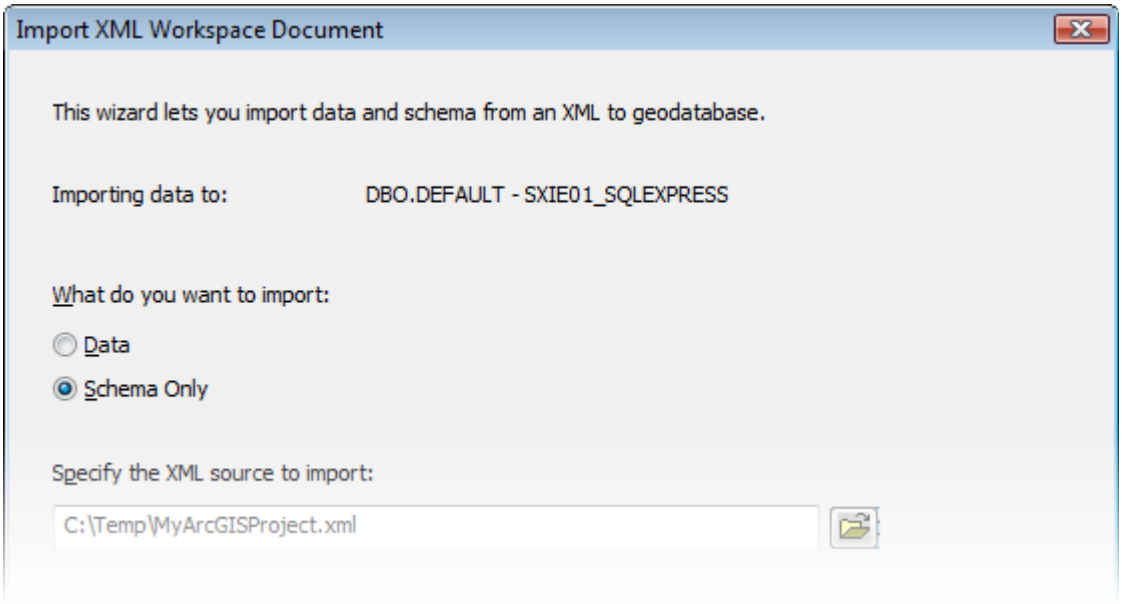

Figure 19: The schema import utility in ArcCatalog.

3. Choose the **Schema Only** option and select the file path of your generated schema.
4. Click **Next**. You will be presented with a summary of the proposed import.

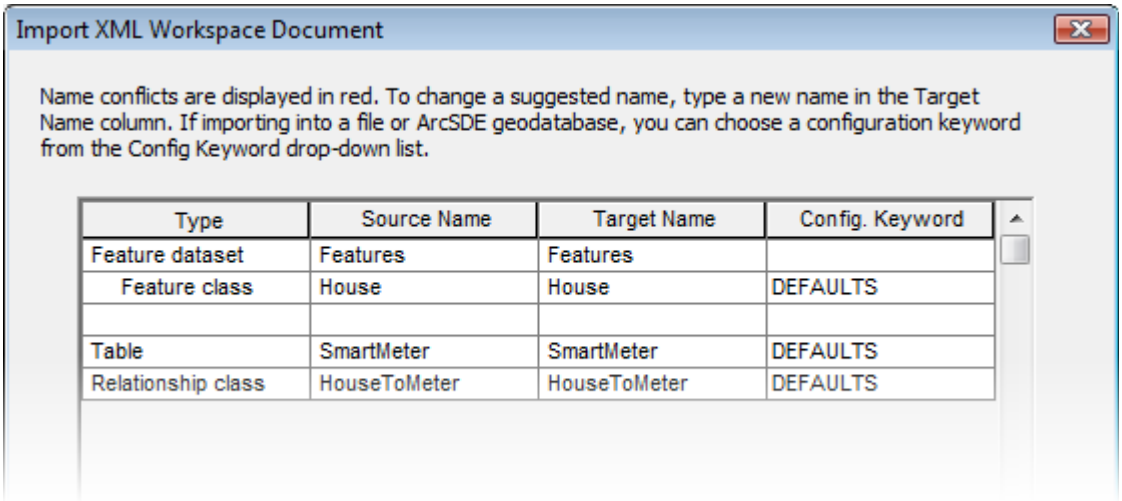

Figure 20: The import summary should contain all modeled Feature Types and Tables.

5. Click **Finish.**
6. The imported schema should be shown in the **Catalog Tree**:

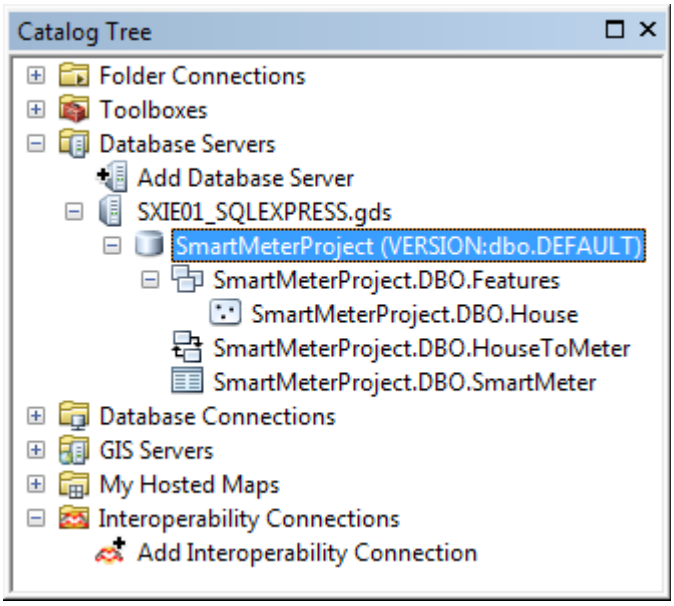

Figure 21: ArcCatalog's Catalog Tree showing the import Smart Meter schema.

7. Viewing the Table Properties, reveals the modeled custom field “HOUSEID”.

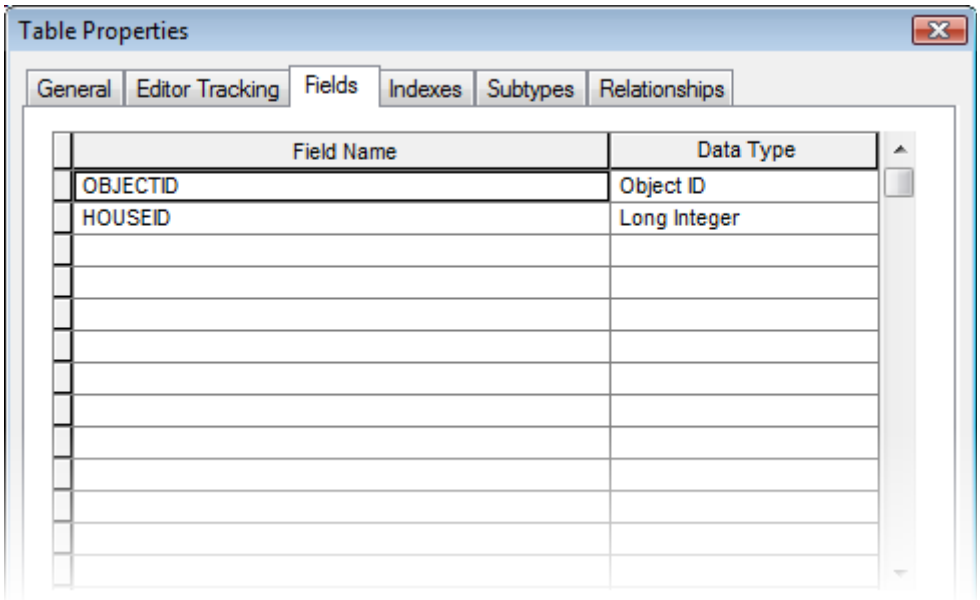

Figure 22: The custom HOUSEID field represented in ArcCatalog.

Summary

In this tutorial, we have modeled a very simple geodatabase consisting of a single Feature Class and a related Table. The Model Pattern for ArcGIS provided a starting structure within which we could model specific ArcGIS concepts. After defining the feature model, it was necessary to configure a Spatial Reference element before generating a valid ArcGIS schema in the form of an XML Workspace document. The final step was to import the generated schema using ArcCatalog's Workspace import wizard.

We have briefly presented how to model a geodatabase from scratch using the UML and Enterprise Architect. One of the key benefits of doing this, however, is the ability to link the geodatabase model to other elements in our modeled system or enterprise architecture. Furthermore, the ArcGIS toolset in Enterprise Architect provides the ability to reverse engineer existing geodatabases into UML models – facilitating better understanding of legacy systems. Both of these topics will be discussed in future tutorials.

Appendix A: Relating ArcGIS concepts to The UML notation

This appendix describes how Enterprise Architect translates ArcGIS concepts, such as Feature Types and Datasets, to UML elements, such as Classes and Packages. Table 1 lists ArcGIS concepts that correspond to UML elements or connectors. Table 2 lists ArcGIS properties that map to element or connector properties, such as a Class Name or Association Role.

Most ArcGIS-specific schema properties, such as *OIDFieldName*, do not have a direct mapping to UML element properties. Instead, these are captured by UML Tagged Values, which form part of Enterprise Architect's UML Profile for ArcGIS. Therefore, a Point is represented as a stereotyped UML Class with a Tagged Value named *OIDFieldName* having a value that specifies the relevant field. The following tables focus on information items not already captured by ArcGIS-specific tagged values.

ArcGIS Concept	UML Equivalent	Stereotype
ArcGIS Workspace	Package	ArcGIS
Feature Dataset	Package	FeatureDataset
Raster Dataset	Package	RasterDataset
Geometric Network	Package	GeometricNetwork
Topology	Package	Topology
Feature Class	Class	Point, Polyline, Polygon, Multipatch
Subtype	Class and Generalization	SubtypeClass and Subtype
Raster Catalog	Class	RasterCatalog
Raster Band	Class	RasterBand
Table (Object Class)	Class	ObjectClass
Coded Value Domain	Class	CodedValueDomain
Range Domain	Class	RangeDomain
Spatial Reference (Coordinate System etc)	Class	SpatialReference
Field	Attribute	Field
Subtype Field	Attribute	SubtypeField
Domain Coded Value	Attribute	DomainCodedValue
Attribute Index	Attribute	AttributeIndex
Spatial Index	Attribute	SpatialIndex
Relationship Class	Association; Association Class	RelationshipClass
Relationship Rule	Association	RelationshipRule
Connectivity Rule	Association; n-ary Association	ConnectivityRule

Table 1: ArcGIS concepts that are represented as UML elements or connectors in Enterprise Architect

Note: Abstract classes are supported by Enterprise Architect's profile, even though there is no direct ArcGIS equivalent. See Appendix E for more information.

ArcGIS Schema Property	UML Field
Feature Dataset Name	Package Name
Feature Class or Table Name	Class Name
Feature, Table or Relationship Class Alias	Class Alias, Association Alias
Relationship Class Name	Association Name
Relationship Class Cardinality	Association Source and Target Role Multiplicity
Relationship Class Backward Path Label	Association Target Role
Relationship Class Forward Path Label	Association Source Role
Relationship Class Notification	Association Direction: esriRelNotificationBackward: Destination → Source esriRelNotificationBoth: Bi-Directional esriRelNotificationForward: Source → Destination esriRelNotificationNone: Unspecified
Field Type	Attribute Type
(Domain Coded Value) Code	Attribute Initial Value

Table 2: ArcGIS schema properties represented as element properties in Enterprise Architect

Appendix B: Creating custom indexes

Setting Indexes on Features Classes and Tables can help to improve performance of a geodatabase. Enterprise Architect supports modeling of Spatial and Attribute Indexes for ArcGIS, using stereotyped UML Attributes.

When you create a new Feature Class or Table, Enterprise Architect creates an Attribute Index that refers to the element's *OBJECTID* field by default. In the case of Feature Classes, a Spatial Index is also created that refers to the *Shape* field. You may modify or delete these Indexes and you may add your own custom Indexes.

You model a custom Attribute Index on a Feature Class or Table as follows:

1. From the ArcGIS toolbox, drag and drop an **Attribute Index** onto the desired Feature Class or Table element.
2. You will be prompted to enter a Name and Type.
3. Name the Index and clear the *Type* field. Click **OK**.
4. Right-click the class element and choose **Attributes** from the context menu. Select the Index from the list of attributes.
5. Click on the Tagged Values tab.
6. Adjust the values of *IsAscending* and *IsUnique* as appropriate.
7. **Select the Index fields:** Click the ellipsis button in the *Fields* Tagged Value. Navigate to the attributes of this class. Select one or more of the UML attributes (ArcGIS fields). Click **OK**.
8. Click Save, then Close.

You model Spatial Indexes using a similar process with the *SpatialIndex* stereotype.

Appendix C: Modeling coded value domains and range domains

Coded Value Domains and Range Domains can be modeled as UML Classes using the ArcGIS profile.

To create a Coded Value Domain:

1. Open the *Domains* diagram under the package named *Domains*.
2. Drag and drop a *Coded Value Domain* stereotype from the ArcGIS toolbox onto the diagram.
3. Enterprise Architect creates a class with stereotype *Coded Value Domain*. It has attributes that define the Field Type, Merge Policy and Split Policy, and it has a sample coded value.
4. **Name the Coded Value Domain:** Open the properties of the class and rename the element appropriately using the Name field.
5. **Specify the Field Type:** Click the **Details** tab, then the **Attributes** button to invoke the **Attributes** dialog. Select the *FieldType* attribute and specify an appropriate esri type in the **Intial Value** field.
6. **Specify the Split and Merge policies:** Select *SplitPolicy* and type an appropriate policy into the **Initial Value** field. Do likewise for the *MergePolicy* attribute.
7. **Add Coded Values:** Use the ArcGIS toolbox to drag and drop the *Domain Coded Value* stereotype onto the *Coded Value Domain* element for as many coded values as are required. This creates a set of UML Attributes on the element, each representing an allowed value.

   When prompted for a name and type, enter the appropriate name for the coded value and clear the **Type** field. (The type is preset for each coded value by the *FieldType* attribute.) Edit the **Initial Value** field for each of these Attributes to specify an appropriate value.

   Figure 23 shows an example of a Coded Value Domain element for a Pipe System.

You create Range Domains by using the *Range Domain* stereotype and a similar process as above, specifying appropriate **Initial Values** for the additional *MinValue* and *MaxValue* attributes.

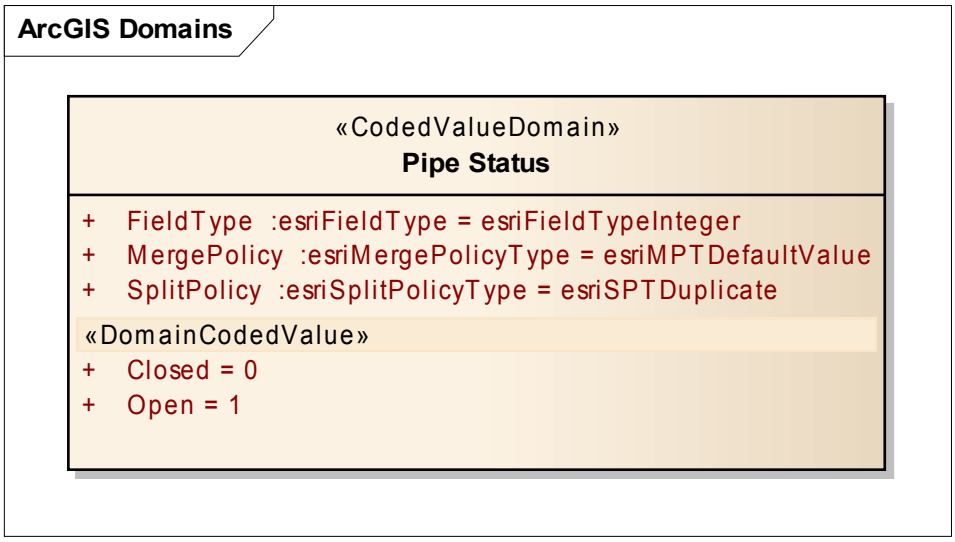

Figure 23: Coded Value Domains are modeled as UML classes. Coded Values are defined by UML Attributes.

Appendix D: Modeling subtypes

ArcGIS subtypes are modeled as stereotyped UML Classes with a Generalization connector to the parent Feature Class. Each subtype contains a subset of the fields from the parent Feature Class. It is uniquely identified among other subtypes of its parent Class by a subtype code. To model subtypes using the ArcGIS Profile:

1. Create an ArcGIS diagram and drop the parent Feature Class onto it.
2. **Create a subtype field in the parent:** A parent Feature Class must have a field – we'll call it *SubtypeCode* – that uniquely identifies its subtypes. To create this field, drag and drop the *SubtypeField* stereotype from the ArcGIS toolbox onto the parent. Name it *SubtypeCode,* or some other meaningful name, and set the **Type** to *esriFieldTypeInteger.*

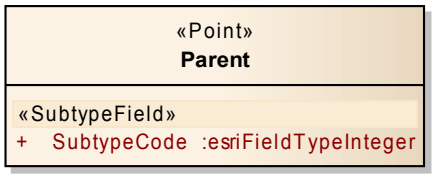

Figure 24: A SubtypeField is defined in the parent Feature class

3. **Create a Subtype Class:** Drag and drop the *Subtype* stereotype from the ArcGIS toolbox onto the diagram. Name the class appropriately.
4. **Assign a unique subtype code:** Each Subtype has an ArcGIS property (UML Tagged Value) called SubtypeCode that specifies its numeric identifier. You need to assign the SubtypeCode a non-negative integer value that will be unique among subtypes of this Feature Class. To do so, edit the *Subtype* properties and change the value of the ArcGIS property, *SubtypeCode*. Note: The SubtypeCode property is initially -1, indicating that a unique ID is not yet assigned.

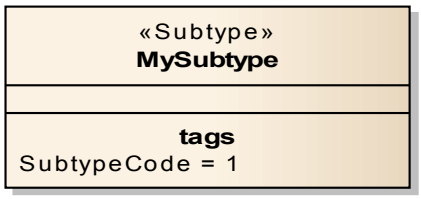

Figure 25: A Subtype is identified by its SubtypeCode.

Alternatively, you could duplicate the SubtypeCode attribute from the parent in each of the Subtypes and assign a unique **Initial Value** for each. This is supported to match an approach used in older CASE tools and is equivalent to using the above SubtypeCode property.

5. **Create Subtype fields:** You can quickly populate a Subtype's fields by copying them from the parent Feature Class. Use the Project Browser to select fields from the parent class and drag them onto the Subtype in the diagram. Now modify the types of the copied fields by editing the UML Attribute properties. Either select a Coded Value Domain (click the ellipsis button next to the **Type** field and navigate to the *Domains* package) or choose a datatype from the droplist.

Note: Each field name in a Subtype must match one of the fields in the parent Feature Class.

6. **Assign a default value for each Subtype Field:** Edit the UML Attribute properties for the Subtype Fields and assign an appropriate **Initial Value**.
7. Repeat steps 3-6 for each Subtype that you require.
8. **Relate Subtype to the parent Feature Class:** For each Subtype, click the *Subtype* relationship on the ArcGIS toolbox and drag from the Subtype to the Feature Class. Alternatively, drag the QuickLinker between classes to create the relationships.
9. **Assign the default Subtype:** Edit the properties of the parent Feature Class and set the **Initial Value** to one of the SubtypeCode values.

Figure 26 shows a simple model of a Feature Class, named *House*, with subtypes *Weatherboard* and *Brick*. The *SubtypeCode* value uniquely identifies the subtypes. The *Material* attribute is typed by an existing Coded Value Domain, *BuildingMaterial,* which must be compatible with esriFieldTypeInteger. In this hypothetical example, we arbitrarily decided that the *LotNumber* field is not relevant to either of these subtypes and therefore omitted it from their defining classes.

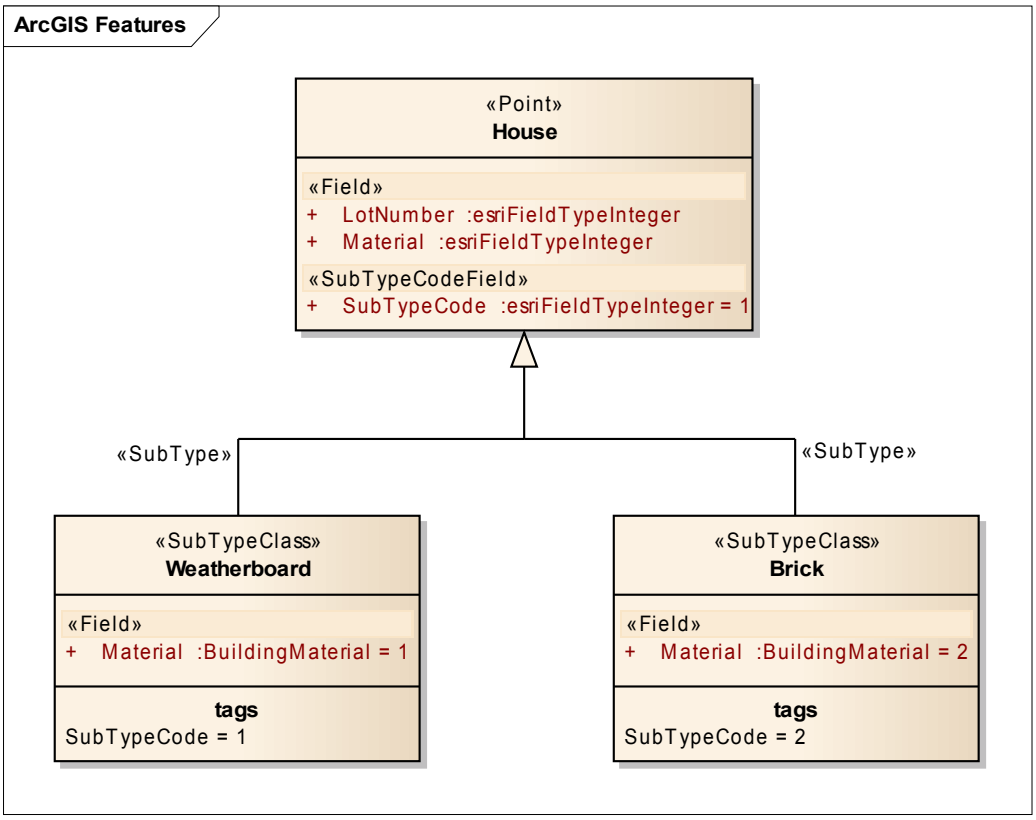

Figure 26: Subtypes are related to the parent Feature Class via UML Generalizations. They contain a subset of the fields in the parent Feature Class. The SubtypeCode must be unique across all subtypes of a given Feature Class.

Appendix E: Modeling abstract classes

One of the fundamental benefits of modeling is that it helps you to manage complexity through multiple levels of abstraction. When designing a geodatabase schema, this applies when several features share the same underlying concepts and may therefore have fields in common. Rather than duplicate those fields in each feature class, you may group them into one or more abstract classes and use UML inheritance to simplify the overall model. The ArcGIS Toolbox includes a UML class, labeled *Abstract Class*, for this purpose.

When you model an *Abstract Class* using the ArcGIS toolbox, Enterprise Architect creates a non-stereotyped UML class with the Abstract property set. Feature Classes that inherit from this class gain all its fields. Thus, when you export the model to an ArcGIS schema file, inherited attributes from the Abstract class are duplicated in each descendant class. (Since there is no equivalent Abstract Class concept in the geodatabase however, the reverse process does not occur when importing an ArcGIS Workspace document.)

The example below shows the effect of an Abstract class on the schema:

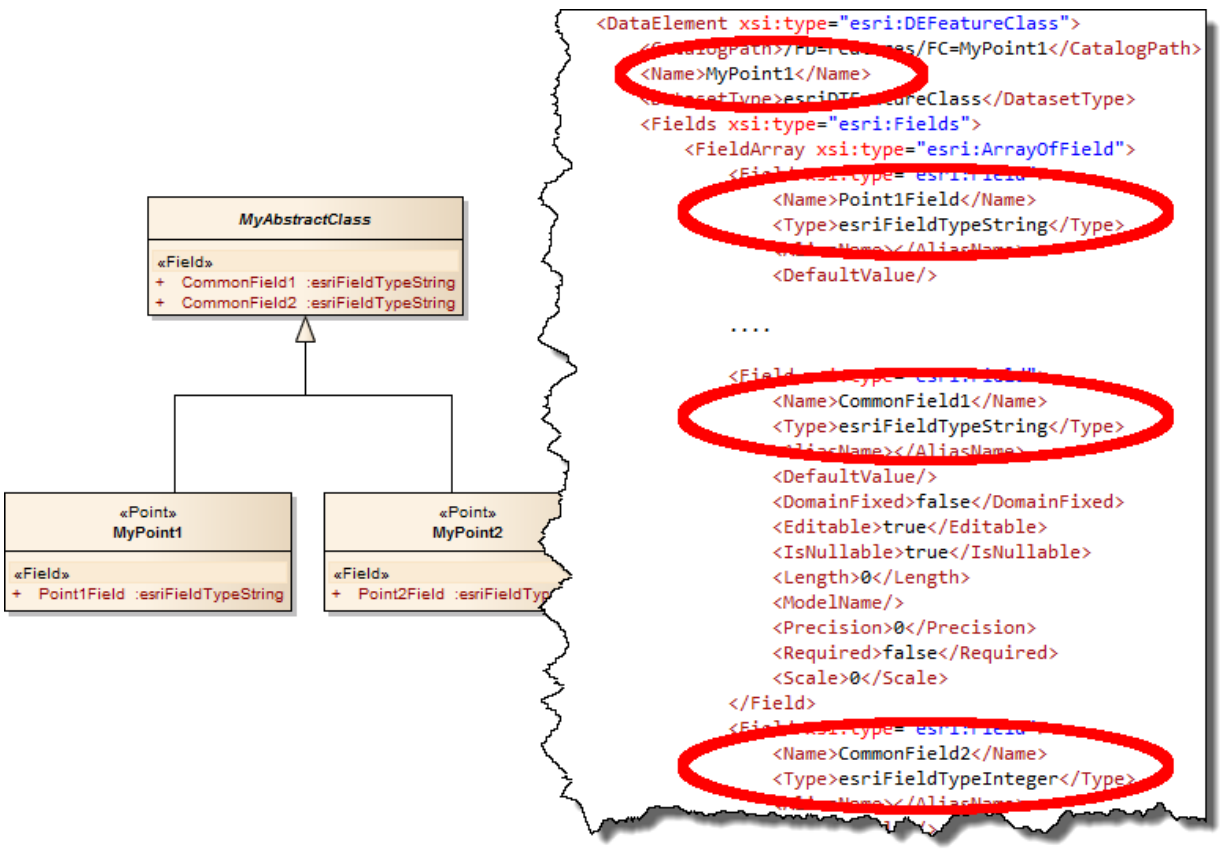

Figure 27: An Abstract Class models a common concept with its fields in a single element. These may be inherited by other Feature Classes.

Multiple *levels* of abstraction are supported by the schema exporter. Thus an Abstract class may inherit from another Abstract Class. Furthermore, *multiple inheritance* is supported. This means a Feature Class can directly inherit from multiple (sibling) Abstract classes.

Appendix F: Customizing "system-level" ArcGIS fields in the model

When you create a Feature Class element using the ArcGIS toolbox, Enterprise Architect automatically creates UML attributes for you that represent the OBJECTID and Shape fields. These attributes have the stereotype "Required Field". Corresponding indexes are also added as stereotyped UML attributes. Similarly, when you create a Table element, an OBJECTID field is added with a corresponding index. Although these "system-level" ArcGIS fields are not shown on diagrams by default, you can edit and display them.

Feature Classes and Tables have an ArcGIS property called *OIDFieldName*, the value of which determines the field that contains the OBJECTID information. By default, *OIDFieldName* points to the OBJECTID UML attribute that is created for you. You can however, reassign the value of *OIDFieldName* property to another field if required. This might be useful if you had to recreate the OBJECTID field, perhaps after having imported model elements from another tool.

To reassign the *OIDFieldName* value:

1. Open the ArcGIS properties for the Feature Class or Table element.
2. Select the *OIDFieldName* property.
3. Click the ellipsis button. This opens a browser window for the Tagged Value. Navigate to the appropriate element and expand its contents. Choose the appropriate attribute.
4. Click **OK**.

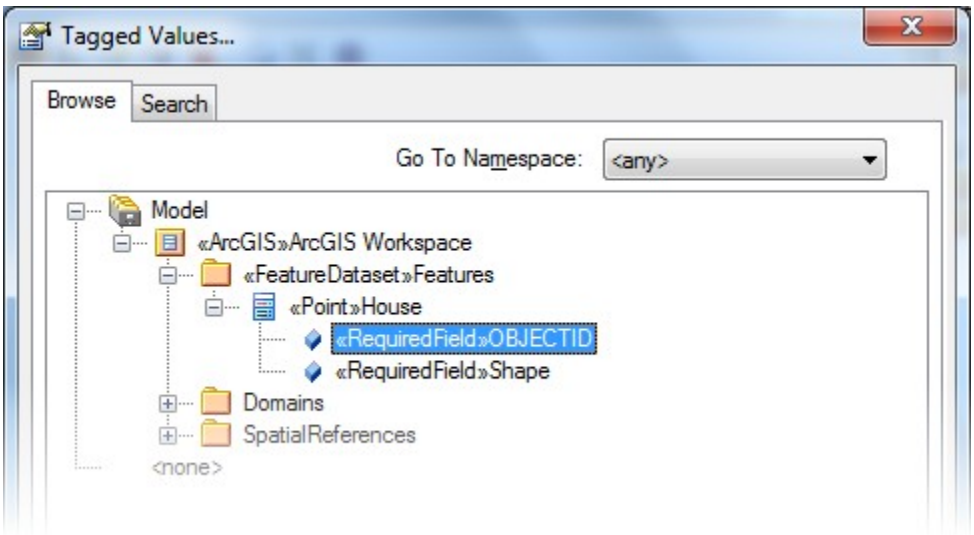

Figure 28: OIDFieldName refers to one of the feature type's attributes.

Reassigning the Shape field of a Feature Class involves a similar process:

1. Open the ArcGIS properties of the Feature Class.
2. Select the *ShapeFieldName* property.
3. Click the ellipsis button. Navigate to the Feature Class and expand its contents. Choose the appropriate Shape attribute.
4. Click **OK**.

Displaying System-level ArcGIS fields on diagrams:

You can control whether the various required fields and indexes are displayed on diagrams. By default, when you create a new ArcGIS element, Enterprise Architect hides fields that are stereotyped with: RequiredField, AttributeIndex or SpatialIndex. This is accomplished via a built-in element display option that lets users suppress UML attributes and operations with specified stereotypes.

The ArcGIS extension for Enterprise Architect makes it easy to override this default setting for selected elements on a diagram. To display or hide system-level settings:

1. Select one or more ArcGIS elements on a diagram. (Ctrl+Left-click to select multiple elements)
2. Choose the menu option: **Extensions > ArcGIS > Show or Hide ArcGIS System Fields**
3. The display of attributes such as OBJECTID, Shape_IDX etc. will be toggled on or off, as shown below.

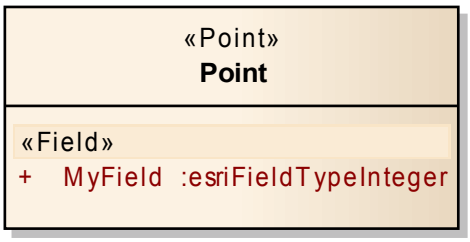

Figure 29: Default behavior: System fields are suppressed when elements are first created on diagrams.

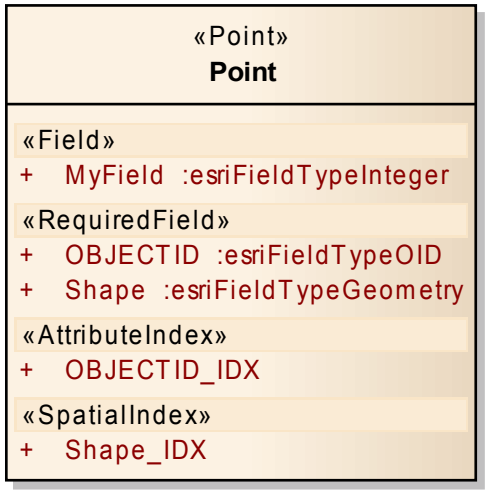

Figure 30: System fields displayed for an ArcGIS profile element