ArcGIS® 9

Introduction to CASE Tools

Copyright © 1999–2004 ESRI.
All rights reserved.
Printed in the United States of America.

The information contained in this document is the exclusive property of ESRI. This work is protected under United States copyright law and the copyright laws of the given countries of origin and applicable international laws, treaties, and/or conventions. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or by any information storage or retrieval system, except as expressly permitted in writing by ESRI. All requests should be sent to Attention: Contracts Manager, ESRI, 380 New York Street, Redlands, CA 92373-8100, USA.

The information contained in this document is subject to change without notice.

CONTRIBUTING WRITERS

Andrew Perencsik, Eddie Idolyantes, Bob Booth, and Julio Andrade.

U.S. GOVERNMENT RESTRICTED/LIMITED RIGHTS

Any software, documentation, and/or data delivered hereunder is subject to the terms of the License Agreement. In no event shall the U.S. Government acquire greater than RESTRICTED/LIMITED RIGHTS. At a minimum, use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in FAR §52.227-14 Alternates I, II, and III (JUN 1987); FAR §52.227-19 (JUN 1987) and/or FAR §12.211/12.212 (Commercial Technical Data/Computer Software); and DFARS §252.227-7015 (NOV 1995) (Technical Data) and/or DFARS §227.7202 (Computer Software), as applicable. Contractor/Manufacturer is ESRI, 380 New York Street, Redlands, CA 92373-8100, USA.

ESRI, ArcView, ArcInfo, ArcCatalog, ArcToolbox, and ArcGIS are trademarks, registered trademarks, or service marks of ESRI in the United States, the European Community, or certain other jurisdictions.

The names of other companies and products herein are trademarks or registered trademarks of their respective trademark owners.

Introduction to CASE Tools

IN THIS DOCUMENT

- **What are CASE tools?**
- **Designing a geodatabase with UML**
- **Generating schema**

The geodatabase brings the physical representations of geographic features closer to their actual real-world counterparts. It is possible to create hydrants, mains, and valves and to define a number of characteristics for each, such as fields, validation rules, relationships, and subtypes. The computer-aided software engineering (CASE) tools subsystem lets you create blueprints of the structure of the geodatabase using a graphical language—the Unified Modeling Language (UML). Using class diagrams, you can represent geodatabase elements, such as feature datasets or geometric networks, and clearly see the relationships among them.

This document discusses how you can use Microsoft® Visio® and Rational Rose to construct your UML and how you can use CASE tools in ArcCatalog™ to generate the schema for your geodatabase. Once you have read this document, read the PDF file Designing_Geodatabases_with_Rational_Rose or Designing_Geodatabases_with_Visio for specific instructions on how to create geodatabase items in UML.

CASE tool functionality does not apply to ArcView® licenses.

What are CASE tools?

There are three general strategies to creating geodatabases. The first two strategies—migrating existing databases to the geodatabase and using tools in ArcCatalog and ArcToolbox™ to create the schema for your geodatabase design—are discussed in *Building a Geodatabase*. This document discusses the third strategy: using UML and the CASE tools subsystem of ArcGIS® to generate the schema for your geodatabase.

UML models are created using specialized tools, such as Microsoft Visio or Rational Software Corporation's Rational Rose, then exported to an intermediate format: the Microsoft Repository or an XML Metadata Interchange (XMI) file. The CASE tools will read the model and allow you to create the geodatabase schema and, optionally, generate code to define custom behavior.

XMI is an Object Management Group (OMG) standard that specifies how to store a UML model in an XML file. ArcGIS now reads models in XMI files as well as models stored in the Microsoft Repository.

Using CASE for schema design and generation

The general strategy for using UML and CASE tools to design and create your geodatabase involves using UML to define all of the schema for the geodatabase, generating that schema, then populating the schema with data. The steps for accomplishing this are outlined below:

1. Create your geodatabase design in UML.
2. Export your UML model to XMI or Microsoft Repository.
3. Use the Schema Wizard in ArcCatalog to create the schema in your geodatabase from your UML model.

Once you have generated the schema, you may want to start directly editing that schema to build your database, but typically you will have existing data with which you want to populate that schema. There are a number of things that can impact performance when loading data into a geodatabase schema, especially when working with network data.

There is more than one strategy for loading data into an existing database schema. Each strategy has its limitations and affects performance of the database. The different strategies for loading your existing data into that schema and the performance considerations of each are outlined in *Building a Geodatabase*.

The general strategy for using UML and CASE tools to design and create your geodatabase involves using UML to define all of the schema for the geodatabase, generating that schema, then populating the schema with data.

Designing a geodatabase with UML

Geodatabase elements, such as tables, feature classes, and relationship classes, follow certain rules that dictate the organizational relationship between elements in the geodatabase relative to each other. For example, feature datasets are defined under the geodatabase, while geometric networks are defined inside feature datasets. Some elements, such as feature classes and relationship classes, can be inside or outside feature datasets.

These aspects of a geodatabase's structure are defined in UML through the use of packages to represent the geodatabase and feature datasets. The UML classes that are used to define feature classes, relationship classes, tables, and geometric networks are defined under these packages.

Tagged values

Tagged values are used to set additional properties of UML elements. For example, you can set the length (in characters) of a string field by using a tagged value.

Tagged values are recognized on several other UML elements: class, attribute, associations, and so on. The following table summarizes the tagged values used for geodatabase schema elements.

Table 1: Tagged values for geodatabase UML elements

Tagged value name	Values/Remarks
Fields:	
Precision	Integer value *Integer fields: number of digits* *Double fields: total number of digits*
Scale	Integer value *Number of decimal places in single and double fields*
Length	Integer value *Width of character fields*
AllowNulls	True/False
Alias	String *The alias name for the field*
Feature class/Object class:	
Geometry type	esriGeometryPoint esriGeometryPolygon esriGeometryPolyline esriGeometryMultipoint *Valid only for feature classes*
Ancillary role	esriNCARNone esriNCARSourceSink *Valid only for junction feature classes*

ConfigKeyword	String value
HasM	True/False *Valid only for feature classes*
HasZ	True/False *Valid only for feature classes*
CLSID	GUID in registry format *If specified, the Schema Wizard will look for a Component Object Model (COM) class identified by the CLSID in the system registry.* *If not specified, the appropriate ESRI (COM) class will be used instead.* *This tagged value will be overwritten by the Code Generation Wizard if code is generated for the* custom feature. *This tagged value can be specified for UML classes representing class extensions as well.*
Alias	String *The alias name for the field*

Geometric network:

GNConfigKeyword	String

Relationship class:

Notification	esriRelNotificationBackward esriRelNotificationBoth esriRelNotificationForward esriRelNotificationNone
IsAttributed	True/False
OriginClass	Name of the origin class
OriginPrimaryKey	Name of the primary key field of the origin class
OriginForeignKey	Name of the foreign key field of the origin class *For 1–1 and 1–M relationship classes, this field lives in the destination class.* *For 1–M/attributed relationship classes, this field lives in the auxiliary <<RelationshipClass>> class.*
DestinationPrimaryKey	Name of the primary key field of the destination class *Valid for M–N/attributed relationship classes only.*
DestinationForeignKey	Name of the foreign key field of the destination class *Valid for M–N/attributed relationship classes only. This field lives in the auxiliary <<RelationshipClass>> class.*

Domain:

Description	String value

Feature datasets

Feature datasets are modeled in UML as stereotyped packages that you create under the geodatabase workspace package. A feature dataset package cannot be created under another feature dataset package; however, other packages created for organizational purposes can. For example, a package that holds all

of the subtypes for a particular class can be created under a feature dataset package.

Geodatabase elements, such as tables, feature classes, and relationship classes, follow certain rules that dictate where these elements are stored in the geodatabase relative to each other. These aspects of a geodatabase's structure are defined in UML through the use of packages to represent the geodatabase and feature datasets.

Feature datasets have a spatial reference associated with them. Spatial references are not modeled in UML. Instead, the spatial reference for a feature dataset is set when generating the schema in ArcCatalog.

Tables, feature classes, and geometric networks

UML classes are used to model feature classes and tables. When schema is generated from the UML model, one table or feature class is created for each class in the model. Each property of the object is mapped to a field of the table or feature class. Required fields are included as properties of the base class and need not be repeated in any inherited classes. For example, you may have a class, Pipes, that has the properties Material and Diameter. If you then create a new class called Mains that is inherited from Pipes, then Material and Diameter are also properties of the Mains class but do not need to be repeated in the Mains class in the UML model.

When schema is generated, each property is mapped as a field on the table or feature class. You can specify the length, scale, and precision of the fields that correspond with these properties in the UML diagram by setting tagged values. You can also use tagged values to set properties for feature classes, class extensions, relationship classes, and interfaces.

When generating schema, all feature classes the wizard creates use the coordinate system of the target feature dataset. Object classes (tables) are created at the workspace level.

The default grid size is 1,000. Grid size is not stored as part of the model because the coordinate system of target feature datasets might be different. You can specify a grid size for each feature class in the Schema Wizard.

Tables are created for those objects that inherit from the Object class. Feature classes are created for those that inherit from the Feature class, while simple or complex network feature classes are created for those that inherit from the Network feature classes.

All network feature classes must be associated with a geometric network (GN). Geometric networks are modeled in UML as special classes. The geometric network and its associated network

feature classes must be created in the same feature dataset package. A junction feature class is created for all features that inherit from Simple Junction or Complex Junction. For more information on geometric networks, see the chapter 'Geometric networks' in *Building a Geodatabase*.

Tables are created for objects that inherit from the Object class, feature classes are created for objects that inherit from the Feature class, and network feature classes are created for objects that inherit from the Network feature class.

Subtypes and domains

Attribute domains are modeled in UML as special classes. When the CASE tool generates the schema from the UML model, these domain classes are stored in the geodatabase as domains. Coded value and range domains, along with their valid values and split and merge policies, are all modeled in this way.

If your tables or feature classes require subtypes, these can also be modeled in UML and be automatically generated when the tables or feature classes are created in the database. Subtypes are modeled in UML as classes related to the parent class through an association stereotyped as Subtype. The Subtype field is specified in the parent class as a stereotype of a UML attribute.

Subtypes are modeled in UML as classes related to the parent class through an association stereotyped as Subtype. You can specify domains and default values for the fields for that subtype.

You can specify default values and domains for each field for a particular feature class or subtype. To associate a domain with a particular field, simply specify the name of the domain as the field type. Similarly, the field's default value can be supplied for feature classes or subtypes by setting the initial value in the UML attribute representing the field.

Attribute domains are modeled in UML as special classes.

If the feature class or table you are modeling does not have subtypes, but you still want to associate domains and default values to fields, you can do so directly on the class object itself.

Relationships

Associations between objects in your UML model are created in the geodatabase as relationship classes. The cardinality of those associations is reflected in the cardinality of the relationship class. The name of the relationship class is the name of the association. The primary and foreign keys are specified directly in the UML model as tagged values of the UML association.

Attributed *relationships* are modeled as classes with the same name as the relationship class, stereotyped as a relationship class. The fields for the relationship class's attributes are modeled in the same way as the attributes of any other class.

Notification direction—the direction messages are passed—for the relationship class can be modeled in UML using tagged values.

Relationship rules apply to subtypes in feature classes and tables. In UML, these relationship rules are modeled as associations between subtypes of the classes participating in the relationship class.

Relationship classes are modeled in UML as associations. The cardinality of the association is reflected as the cardinality of the relationship class. Attributed relationships are modeled as classes with the same name as the relationship class.

Composite relationships are modeled in UML as *aggregation*. In a composite relationship, the origin class is always the parent in the aggregation. Composite relationship classes always have a one-to-many relationship.

To learn more about relationship classes, see the chapter 'Defining relationship classes' in *Building a Geodatabase*.

Composite relationship classes are modeled in UML as aggregations between classes.

Connectivity rules

Connectivity rules apply to network feature classes and subtypes of network features participating in the same geometric network. These connectivity rules are modeled in UML as a special association stereotyped as *ConnRule*.

For *edge–junction connectivity rules*, the ConnRule association is between the edge subtype and the junction subtype. The junction in one of the edge–junction rules can be set as the default junction by stereotyping the association end as Default. *Edge–edge rules* are represented by a UML N-ary association that involves two edge subtypes and any number of junction subtypes. One of the junction subtypes must be marked as the *default junction* by stereotyping the association end as Default.

Edge–Edge rules are represented by a UML N-ary association. One of the junction classes or subtypes must be marked as the default junction by stereotyping the association end as Default.

Generating schema

The CASE tools subsystem of ArcGIS has two parts: the Code Generation wizard and the Schema wizard.

For more information on the ESRI® object model and on generating code for your custom objects using the Code Generation Wizard, see *Modeling Our World* and *Exploring ArcObjects*.

Semantics checker

The CASE tools expect UML models to be created following a set of modeling rules. For example, a network feature class must be associated with a geometric network. The semantics checker can be used to verify that a model stored in the Microsoft Repository or XMI has been correctly defined. It will produce a report with the list of errors encountered in the model. You should use the semantics checker before running the CASE tools wizards. You can run the semantics checker from within Visio when the template diagram is loaded.

Applying your model to existing data

You can update the schema of a geodatabase with information stored in an object model. For example, you can import data to a geodatabase, then apply a model to add subtypes, relationships, and other elements. Alternatively, your current schema could have been created previously with the Schema Wizard based on a model. Since then, the model might have changed, and you may want to update the database.

You can use the Schema Wizard to modify an existing geodatabase, even if data has already been loaded. To do so, select the target geodatabase and run the Schema Wizard. When the wizard starts, some elements in the model are searched for and matched to objects in the database—feature classes and tables, for example. If found, the wizard will modify them. The Schema Wizard will show the matched elements with a red shadow in the tree view.

The Schema Wizard will show matched elements with a red shadow in the tree view.

For a number of reasons, your database may contain feature classes and fields whose names do not correspond exactly to the names in the UML model. Perhaps you have applications written previously that rely on the existing field's name being different. For example, a field could be named Hgt in the database and Height in the model. The geodatabase lets you assign a model name to feature classes and fields, and the CASE tools use the model name when matching them to UML elements. Some properties of the matched object may be read-only—for example, the spatial reference of an existing feature dataset.

Conversely, other properties of the existing object will be changed based on the information in the model, such as the domain assigned to a field. In the former case, a locked database icon will appear in the form or tab displaying the particular property. In the latter case, a model icon will be used instead.

Those properties that cannot be changed when reapplying the model will have a locked database icon on the right side of their properties dialog box such as the top image above, while those that can be modified will have a model icon such as the bottom image above.

Since the schema of matched elements is modified when a model is applied, exclusive schema locks are acquired for the elements that will be modified. These locks can be established only if there are no other users connected to the database and the current user has the right permissions. Since the schema is modified, you should back up your database before applying the changes.

Each geodatabase element supported by the Schema Wizard follows a set of rules when the model is reapplied. The following is a description of these rules:

Feature datasets: The Schema Wizard uses the name of the UML package representing the feature dataset to search for an existing feature dataset in the target database. The spatial reference and spatial domain properties become read-only.

Other model elements are defined inside feature datasets, such as geometric networks and feature classes. These elements will be found in the existing database only if the feature dataset itself is found.

Feature classes: The name of the UML class representing the table or feature class will be used to search for an existing object in the geodatabase. The comparison is first made against the model names of the existing feature classes. If no match is found, the comparison is made against the names of the existing feature classes. If still no match is found, you can manually set the match using the Exists tab in the feature class's properties dialog box.

Only feature classes with the same feature type can be used when matching manually. Also, only feature classes in the same location are available for matching—for example, the feature classes under the matched feature dataset.

When the Schema Wizard is run, it updates the model name of both existing and new feature classes and fields. This ensures the matching will occur automatically the next time the model is applied.

If no match between the model and the database is found for a feature class or table, you can manually set the match using the Exists tab in the feature class properties dialog box.

If the UML class contains additional fields, these can be marked for addition in the Exists tab of the feature class properties dialog box.

Fields: As with feature classes, fields are matched based on model name, and a manual match can be done. The UML class may also contain additional fields. These can be marked for addition in the Exists tab of the feature class properties dialog box. Existing fields not matched are left untouched. Reapplying the model to existing data does not drop fields.

For matched fields, the field type, length, precision, and scale are read-only properties. However, the domain and initial value will be updated if they have been set in the model (the domain must have the same field type).

Domains: Domains are matched based on the name of the UML class and the existing domain. If found, the domain will be altered with modifications made to the UML class. For example, new codes can be added to a coded value domain.

Subtypes: Subtypes of matched feature classes are deleted and re-created using the subtypes in the model. Because connectivity and relationship rules are created among subtypes, they are deleted and re-created as well. Whenever you use ArcCatalog to add a subtype or rule, you should make sure the model is updated accordingly in case you ever reapply it.

Relationship classes: The name of the relationship class in the model will be used to search for an existing relationship class in the target database. Existing relationship classes that are not matched will be left untouched. The policies on modifying vary, depending on the type of relationship class.

Nonattributed relationship classes are deleted and re-created. Along with the relationship class, the relationship rules are deleted and re-created as well.

Internally, attributed relationship classes are implemented by creating an extra table in the database. This table holds attributes of the relationships and keys to the rows in the related tables. Because this table may be holding data already, only the relationship class's rules, not the relationship class itself, are deleted and re-created.

Geometric networks: The name of the class representing the network will be used to search for an existing geometric network in the target database. As mentioned before, a geometric network is matched only if the feature dataset it belongs to is matched as well. All existing connectivity rules are deleted and re-created.