Extending the Geodatabase with Custom Objects

Erik Hoel

Brian Goldin

Goals

- Develop an understanding of
  - ArcInfo 8 Geodatabase
  - non-programmatic customization opportunities
  - how to program custom objects
- How to proceed forward
  - other UC'00 sessions
  - literature

Agenda

- ArcInfo 8 Geodatabase
  - review
  - non-programmatic customization
- Programming custom objects
  - general process
  - important interfaces
  - common navigation

ArcInfo 8 Geodatabase

ArcInfo 8 Geodatabase

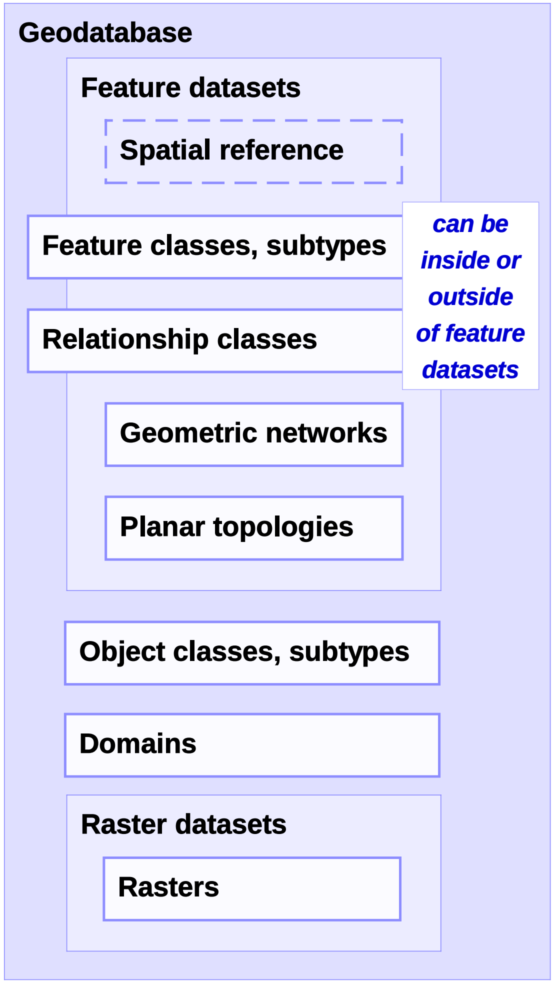
- A new object-oriented geographic data model
- All relational data storage using ArcSDE
- Versioning and long transactions
- New data access objects for application software developers
- Component based technology for developing custom objects and features

ESRI

New Features at 8.1

- Dimension features
- Enhanced support for custom features in the editor
- Dynamic segmentation
- Direct import/export of geodatabase data
- New connectivity rule
- CASE tools enhancements
- Performance enhancements

Geodatabase Elements



- Objects, object classes
- Features, feature classes
- Relationships, relationship classes
- Geometric networks
- Feature datasets
- Validation rules, domains
- Spatial references
- Rasters and other dataset types in the future

Objects

A table stores an ObjectClass

A row stores an Object

OWNER

OID	Name	Address	. . .
518	Bob	38 Oak St.	

- Objects: entities with properties and behavior
- An object is an instance of an object class
- All objects in an object class have the same properties and behavior
- An object can be related to other objects via relationships

Features

FeatureClass (table)

PARCEL

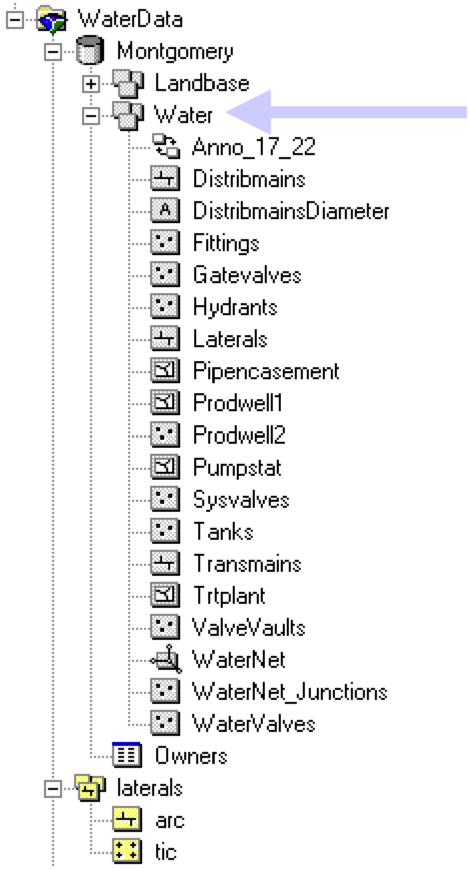
Feature (row)

OID	Shape	Type	. . .
524	X,Y,Z,M, ...	Private	. . .

- A feature is a spatial object
- Features have location
  - a spatial attribute of type geometry
- Features can participate in network and topological relationships
- A feature class is an object class that stores spatial objects (features)
- All features in a feature class have the same spatial reference

Feature Datasets

- Container for feature classes
  - shared spatial reference
- Analogous to a coverage
  - less restrictive
- May also contain
  - relationship classes
  - geometric networks



- WaterData
  - Montgomery
    - Landbase
    - Water
      - Anno_17_22
      - Distribmains
      - DistribmainsDiameter
      - Fittings
      - Gatevalves
      - Hydrants
      - Laterals
      - Pipencasement
      - Prodwell1
      - Prodwell2
      - Pumpstat
      - Sysvalves
      - Tanks
      - Transmains
      - Trtplant
      - ValveVaults
      - WaterNet
      - WaterNet_Junctions
      - WaterValves
    - Owners
  - laterals
    - arc
    - tic

Validation Rules

- Store attribute, connectivity and spatial rules on objects as part of the geodatabase
- Pre-defined, parameter driven:
  - attribute range rule
  - attribute set rule
  - connectivity rule
- Perform custom validation by writing code

Domains

- Describe the legal values of a field type
  - used to ensure attribute integrity
- Can be shared among classes
- Uniquely named
- Types of domains
  - range
    - a tree can have a height between 0 and 300 feet
    - a road can have between 1 and 8 lanes
  - coded value (e.g., a set)
    - a tree can be of type oak, redwood, or palm
    - a road can be made of dirt, asphalt, or concrete

Subtypes

- Partition the objects in an object class into like groups
- Defined by the value of a subtype code field
- All subtypes:
  - have the same attribute schema
  - have the same behavior schema
  - can have different default values and domains for each field

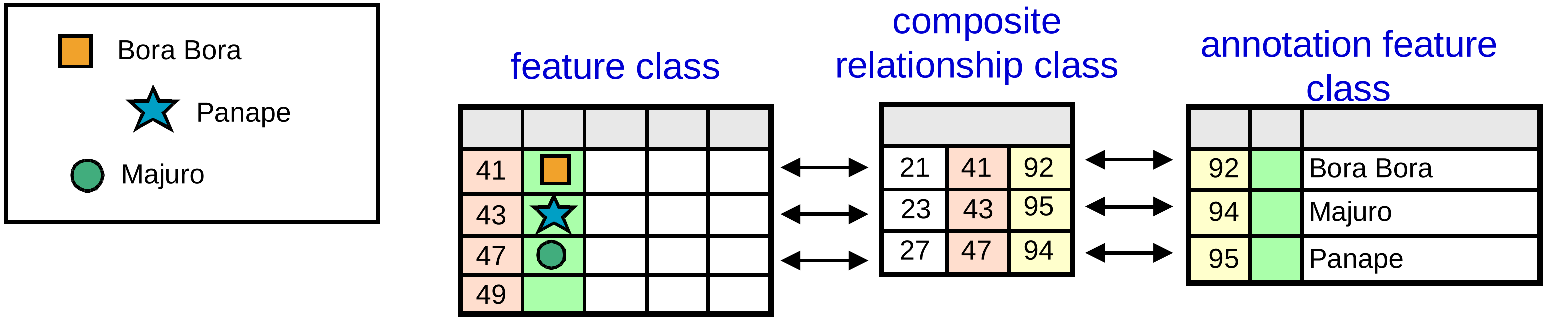
fid	geom	subtype	width	lanes	name
101		asphalt	85.3	4	Chimayo Highway
102		concrete	45.1	2	Acequia de Isabel
103		asphalt	75.9	4	Calle Petra
104		gravel	35.2	2	Maximilian Road

Relationship Classes

- A relationship class is an association between two object classes
- Relationship classes may be 1:1, 1:n, n:m
- An object class may participate in multiple relationship classes
- Related objects can message each other
  - origin to destination, destination to origin, both, neither
  - can trigger behavior (cascade delete, move to follow, custom…)

Annotation

- An example of a graphic feature class
- Annotation feature classes may be
  - feature-linked
  - non feature-linked
- Composite relationship manages link
- Can store text as well as other graphics
  - lines, arrows, boxes, etc.



■ Bora Bora
★ Panape
● Majuro

feature class

41	■			
43	★			
47	●			
49				

composite relationship class

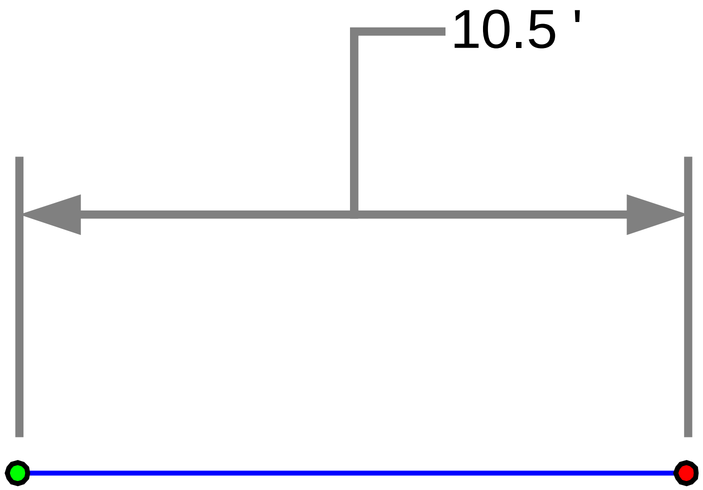
21	41	92
23	43	95
27	47	94

annotation feature class

92		Bora Bora
94		Majuro
95		Panape

Dimension Features

- Type of annotation that displays specific distances on a map
- Stored in a dimension feature class
- Graphic feature
- “Smart” feature
  - – special drawing
  - – special editing

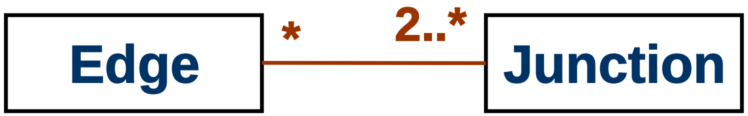


Geometric Networks

- Used to model network systems
- Topological relationship between feature classes
- Each feature class has a topological role in the network (i.e., junction or edge)
- A network may have multiple feature classes in the same topological role
- Topology based upon geometric coincidence, always live
- Feature classes must be in the same feature dataset

Network Feature Classes

- Network features live in a geometric network
- Directly support network analysis
- Types:
  - simple junction
  - simple edge
  - complex junction
  - complex edge



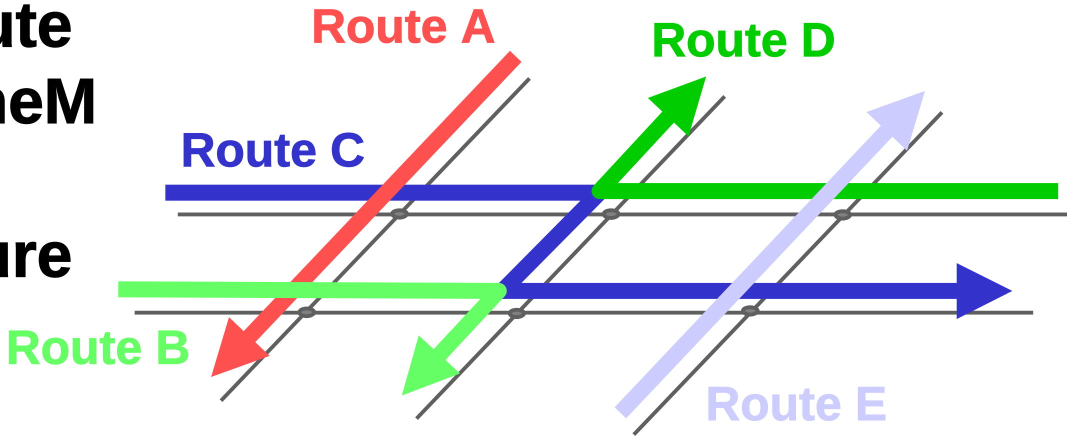
- Integrity constraint:
  - edge must have a junction at each endpoint

Connectivity Rules

- Help you maintain a valid network
- Constrain permissible connectivity
  - default GN behavior allows any edge to connect to any junction
- Connectivity rules include:
  - edge-junction rules
    - cardinality
  - edge-edge rules
    - permissible junction types
    - default junction type

Dynamic Segmentation

- True dynamic segmentation (DynSeg)
  - display table or route events as layer in Map
  - interactively find a location along a route
- Event tables can be INFO, DBASE, Geodatabase, or OLE DB
- Route data can be coverage route system, PolyLineM Shapefile, or PolyLineM feature class

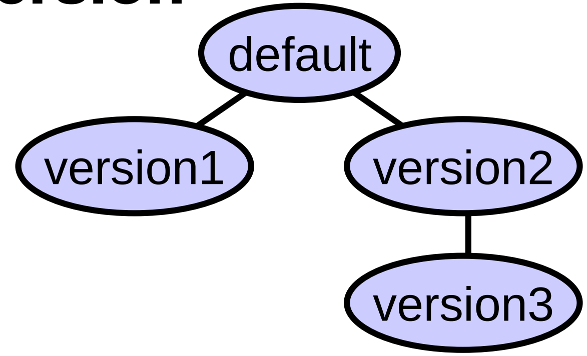


Planar Topology

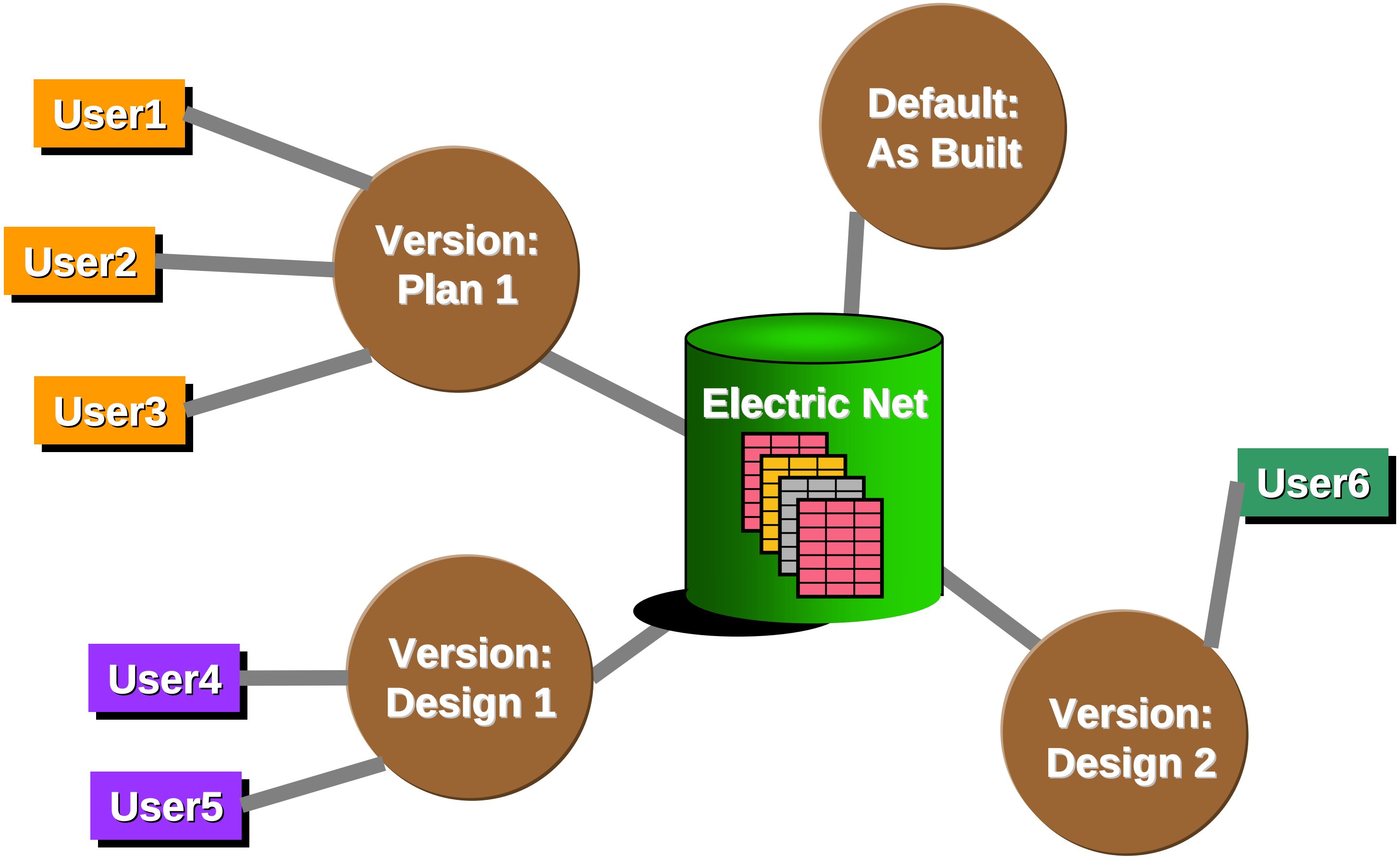
- Feature classes in an integrated feature dataset participate in a planar topology
- Features share boundaries
- Editor tools allow you to edit and maintain shared boundaries
- Use the *Integrate* command in the Editor to ensure coincident boundaries
- Use shared edge edit tool to edit shared boundaries and maintain topological relationships

Versions

- Object classes, feature classes, relationship classes, geometric and logical networks may all be versioned
- A version spans all multi-versioned objects in the database
- Schema is constant across all versions
- Versions differ only in those features or rows or elements modified in each version
- A user can connect to and work with any version of the database - majority will work with the Default version

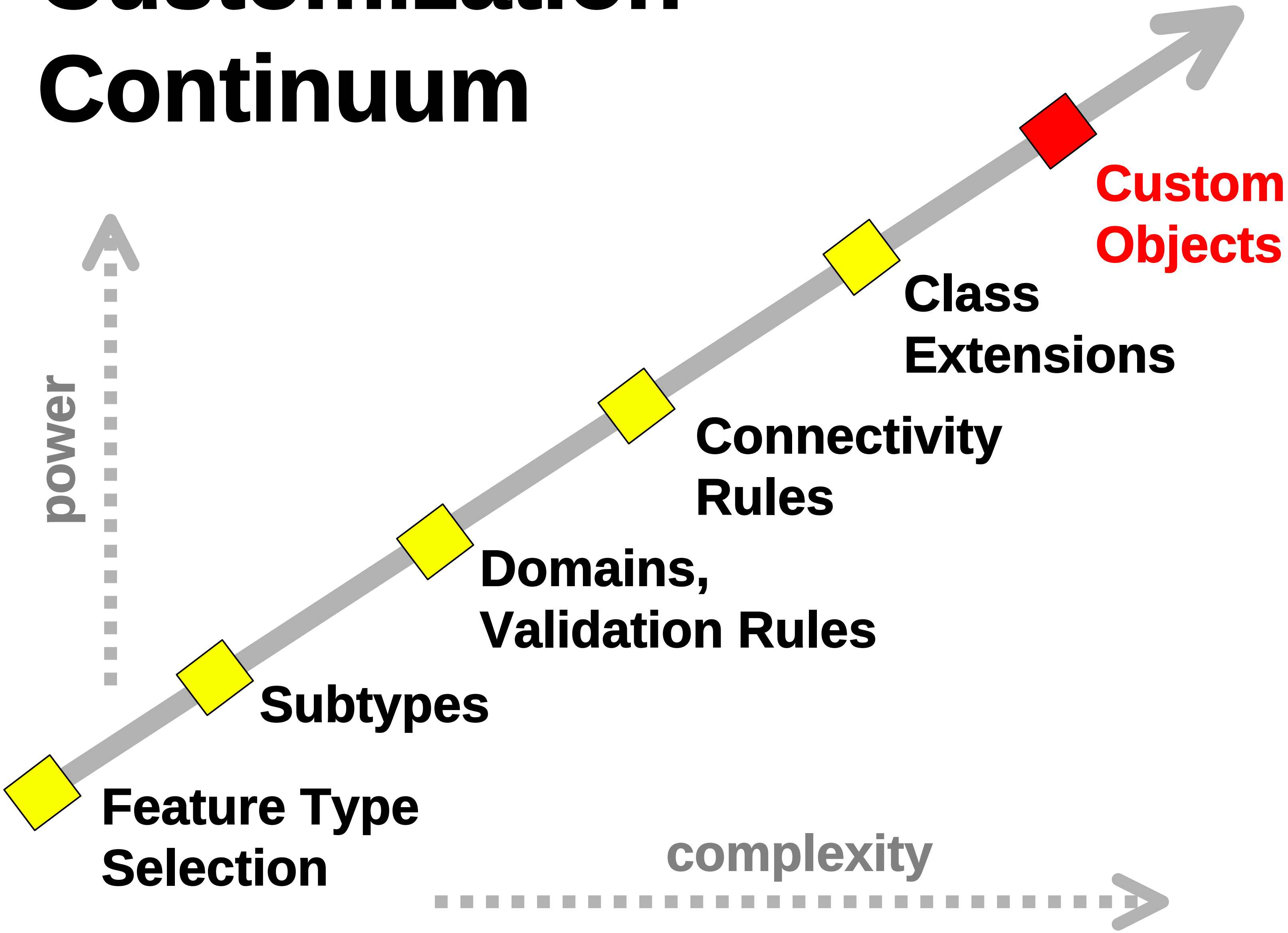


Multi-Versioned Database



Geodatabase Customization

Customization Continuum



Customizing Existing Classes

- Import an object class
  - import template from another object class in any Geodatabase
- Define a new feature class (object class)
  - add attribute fields, set geometry type, spatial reference, etc.
- Edit the behavior of an object class
  - set subtypes, domains, relationships, etc.

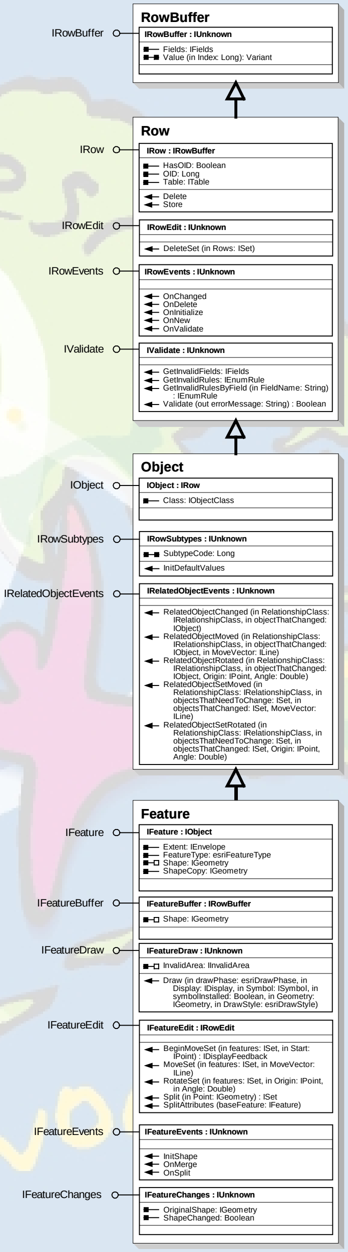
Modeling Additional Behavior

- System can usually be customized without writing custom behavior
- If it is necessary to create additional custom behavior on the object or class
  - nearly any COM compliant language can be used: VC++, VB, Delphi
  - CASE tools and ESRI Code Generation and Schema Wizards make this a lot easier

Class Extensions

- Non-spatial, table-centric customization
- Extension of the object class
  - not a subclassing of an object or object class
- Appropriate for:
  - storing class variables (C++ static variables)
  - custom validation
  - custom property inspectors
  - class level behavior
  - related object creation events
  - class description

Custom Objects



RowBuffer

IRowBuffer

IRowBuffer : IUnknown
- Fields: IFields
- Value (in Index: Long): Variant

Row

IRow

IRow : IRowBuffer
- HasOID: Boolean
- OID: Long
- Table: ITable
- Delete
- Store

IRowEdit

IRowEdit : IUnknown
- DeleteSet (in Rows: ISet)

IRowEvents

IRowEvents : IUnknown
- OnChanged
- OnDelete
- OnInitialize
- OnNew
- OnValidate

IValidate

IValidate : IUnknown
- GetInvalidFields: IFields
- GetInvalidRules: IEnumRule
- GetInvalidRulesByField (in FieldName: String) : IEnumRule
- Validate (out errorMessage: String) : Boolean

Object

IObject

IObject : IRow
- Class: IObjectClass

IRowSubtypes

IRowSubtypes : IUnknown
- SubtypeCode: Long
- InitDefaultValues

IRelatedObjectEvents

IRelatedObjectEvents : IUnknown
- RelatedObjectChanged (in RelationshipClass: IRelationshipClass, in objectThatChanged: IObject)
- RelatedObjectMoved (in RelationshipClass: IRelationshipClass, in objectThatChanged: IObject, in MoveVector: ILine)
- RelatedObjectRotated (in RelationshipClass: IRelationshipClass, in objectThatChanged: IObject, Origin: IPoint, Angle: Double)
- RelatedObjectSetMoved (in RelationshipClass: IRelationshipClass, in objectsThatNeedToChange: ISet, in objectsThatChanged: ISet, MoveVector: ILine)
- RelatedObjectSetRotated (in RelationshipClass: IRelationshipClass, in objectsThatNeedToChange: ISet, in objectsThatChanged: ISet, Origin: IPoint, Angle: Double)

Feature

IFeature

IFeature : IObject
- Extent: IEnvelope
- FeatureType: esriFeatureType
- Shape: IGeometry
- ShapeCopy: IGeometry

IFeatureBuffer

IFeatureBuffer : IRowBuffer
- Shape: IGeometry

IFeatureDraw

IFeatureDraw : IUnknown
- InvalidArea: IInvalidArea
- Draw (in drawPhase: esriDrawPhase, in Display: IDisplay, in Symbol: ISymbol, in symbolInstalled: Boolean, in Geometry: IGeometry, in DrawStyle: esriDrawStyle)

IFeatureEdit

IFeatureEdit : IRowEdit
- BeginMoveSet (in features: ISet, in Start: IPoint) : IDisplayFeedback
- MoveSet (in features: ISet, in MoveVector: ILine)
- RotateSet (in features: ISet, in Origin: IPoint, in Angle: Double)
- Split (in Point: IGeometry) : ISet
- SplitAttributes (baseFeature: IFeature)

IFeatureEvents

IFeatureEvents : IUnknown
- InitShape
- OnMerge
- OnSplit

IFeatureChanges

IFeatureChanges : IUnknown
- OriginalShape: IGeometry
- ShapeChanged: Boolean

Motivation

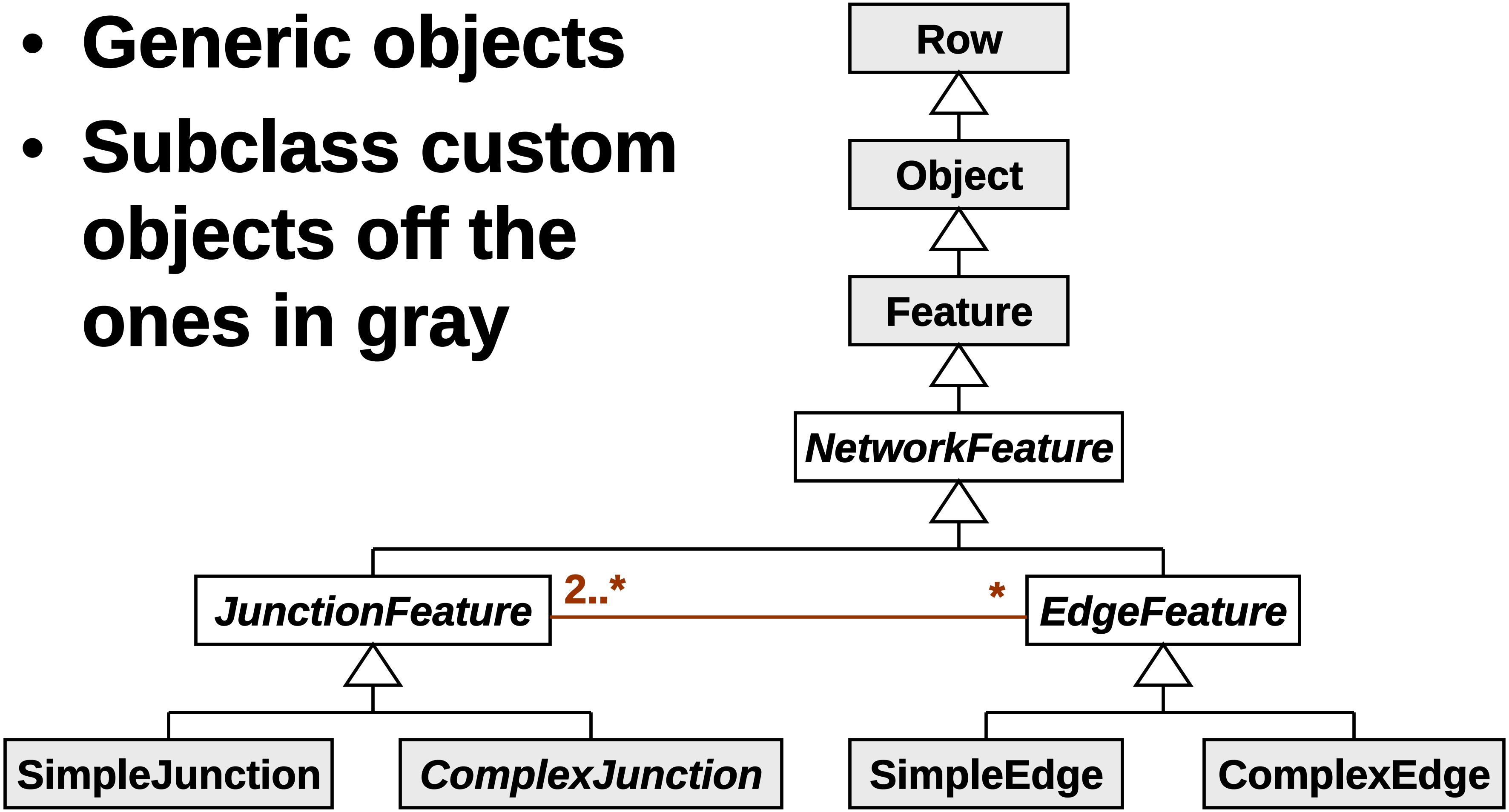
- Used for the most aggressive of customizations
  - feature linked annotation
  - dimension features
- Sometimes custom behavior cannot be supported in the class extension
  - custom notifications
  - linkages to foreign data sources
  - caching properties between objects

Developing Custom Objects

- Custom objects requires programming in COM-compliant language
  - only VC++ or Delphi (COM aggregation issue)
  - class extensions can also use VB or VJ++
- CASE tools and ESRI Code Generation Wizard makes it easier
  - generates an ATL-based VisualStudio project with stubbed methods

Geodatabase Objects

- Generic objects
- Subclass custom objects off the ones in gray



Custom Objects

- Developers create custom objects and complex data schemas
- Semantically, no difference between ESRI supplied and developer-supplied custom objects
  - merely support required interfaces
  - augment with new interfaces consumed by your apps and clients

Custom Objects

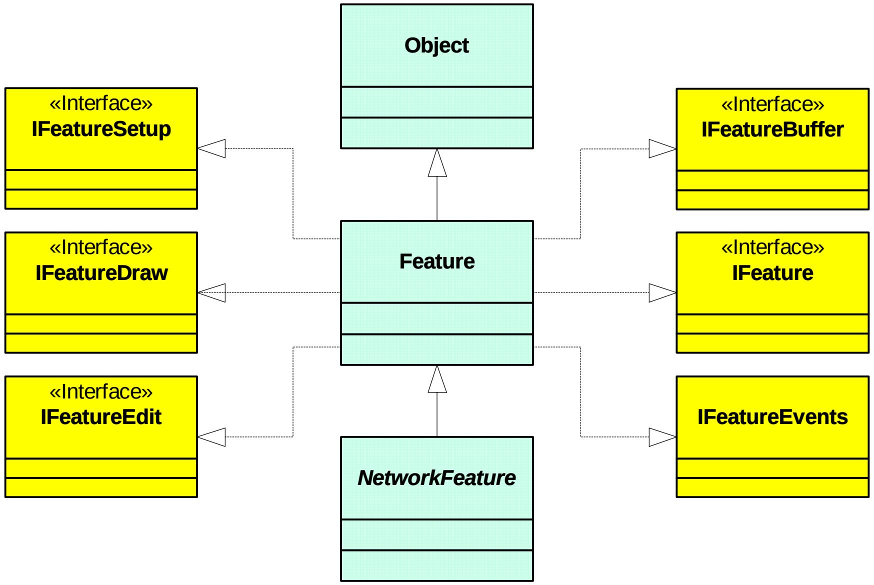
- What you will need
  - UML and Repository aware CASE tool
    - Visio Enterprise
  - Visual C++
  - Geodatabase data model diagram
  - ArcCatalog
  - OO programming skills and knowledge of COM (ATL a big plus)

Creation Process

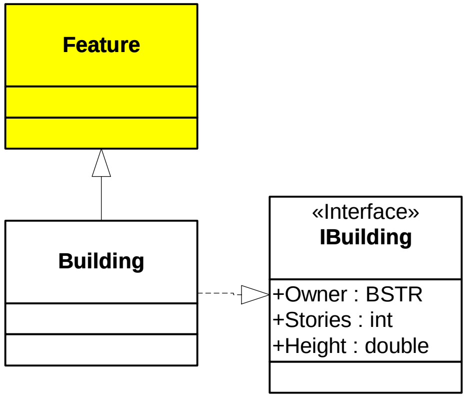
- Create the object model
  - 3rd party CASE tool
- Export to the Microsoft Repository
  - 3rd party CASE tool UML export wizard
- Generate stub-code
  - ESRI supplied wizard (VC++ only)
- Implement custom behavior
  - you program the stubbed methods
- Create the Geodatabase schema

Creation Process

- Base it on a Geodatabase object
  - give it custom behavior, properties



ESRI provides this...



You do this...

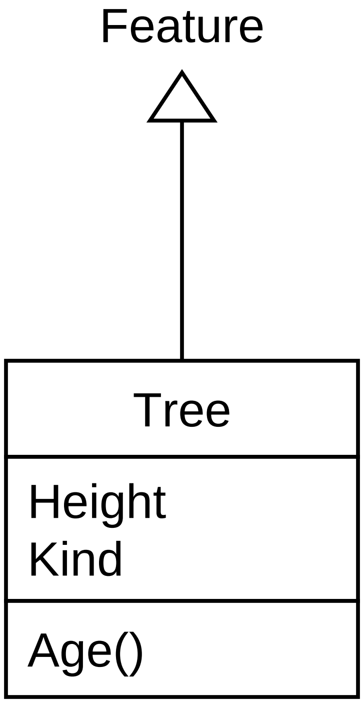
Conceptual Example

Feature

Object Model

COM Implementation

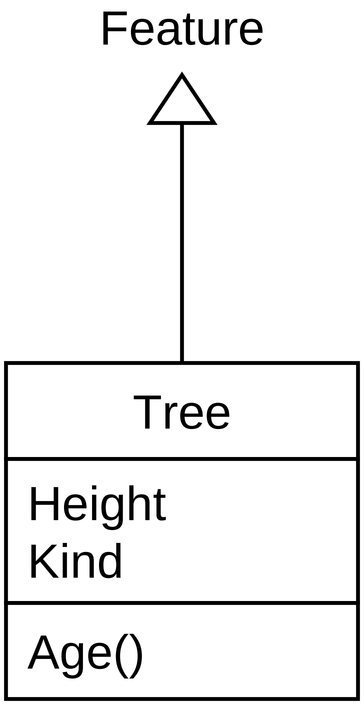
Conceptual Example



Object Model

COM Implementation

Conceptual Example



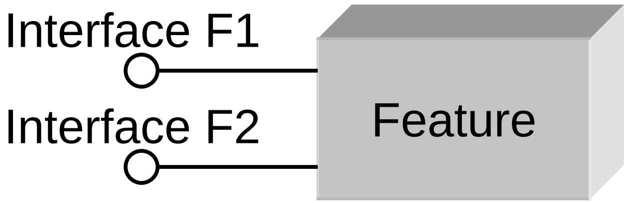
Feature

Tree

Height
Kind

Age()

Object Model



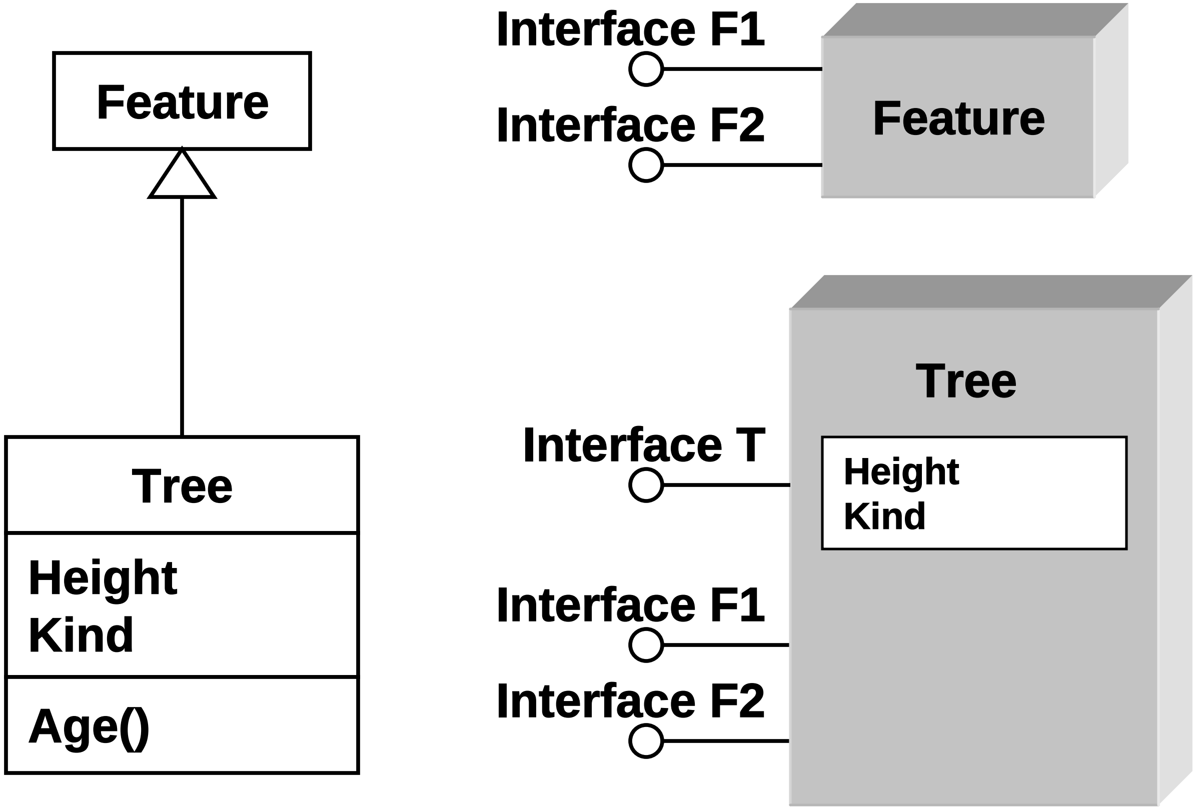
Interface F1

Interface F2

Feature

COM Implementation

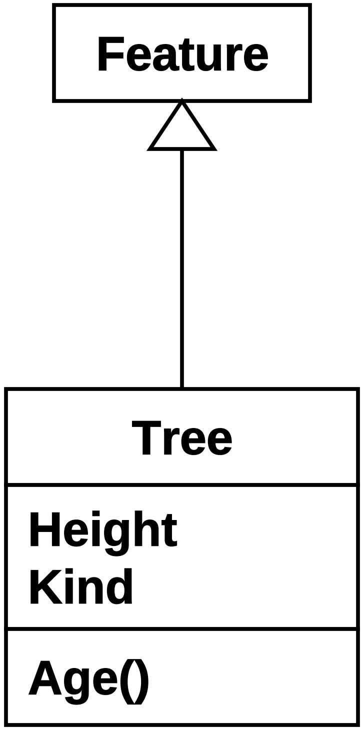
Conceptual Example



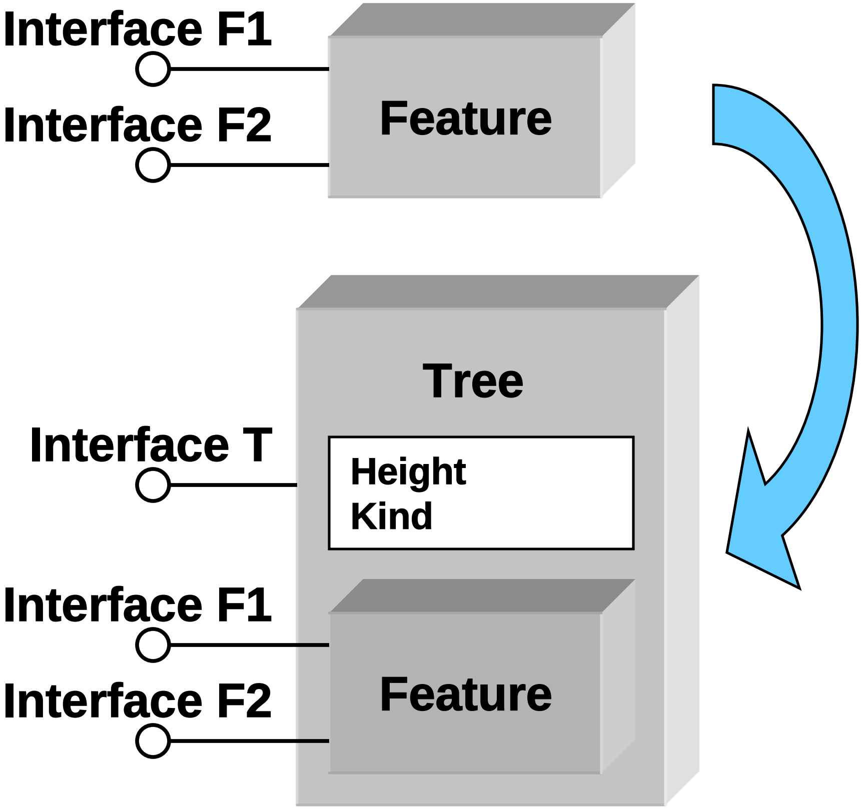
Object Model

COM Implementation

Conceptual Example



Object Model



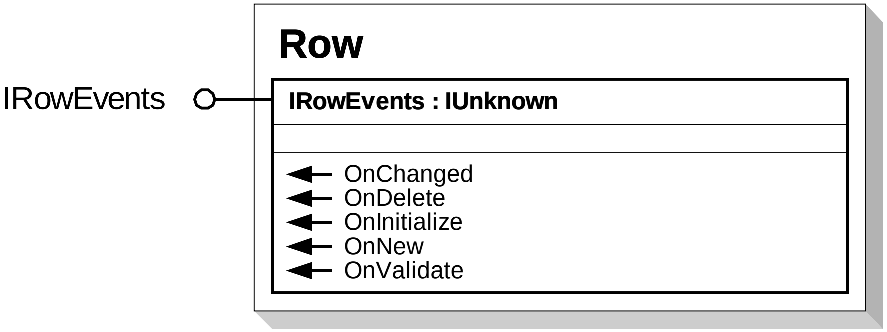
COM Implementation

Programming Custom Objects

Programming Custom Objects

- Developers will typically
  - override methods on I*Event interfaces
  - add new interfaces
  - occasionally override other interfaces (e.g., IFeatureDraw)
- Custom objects and class extensions are often developed as a pair
- Modest collection of interfaces and components to pay particular attention to

IRowEvents



- Standard events
  - OnChanged, OnDelete, OnNew, …
- Good hooks for triggering behavior
- Generic behaviors are NOOPs
  - QI to enclosing outer
- Returning bad HRESULT aborts current edit operation

IRowEvents

- OnNew
  - called in context of Store()
  - after object added to cached collection of new and updated objects
  - before related object classes are notified of object's creation
- OnInitialize
  - called on existing objects
  - after the row has been setup (i.e., property values hydrated)
  - use this event to reset local member variables

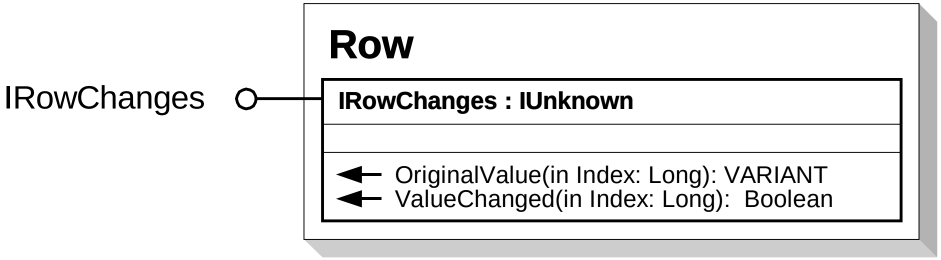
IRowEvents

- OnChanged
  - called in context of Store() on existing object
  - after weights and enabled/disabled pushed to logical network (only on network features)
  - after object added to cached collection of new and updated objects
  - before related object classes are notified of object’s modification

IRowEvents

- OnDelete
  - called in context of Delete() or DeleteSet()
  - called as a side effect of network operations that result in the deletion of a network feature
  - before related part objects (r.e., composite relationships) are deleted
  - before relationship instances are deleted
- OnValidate

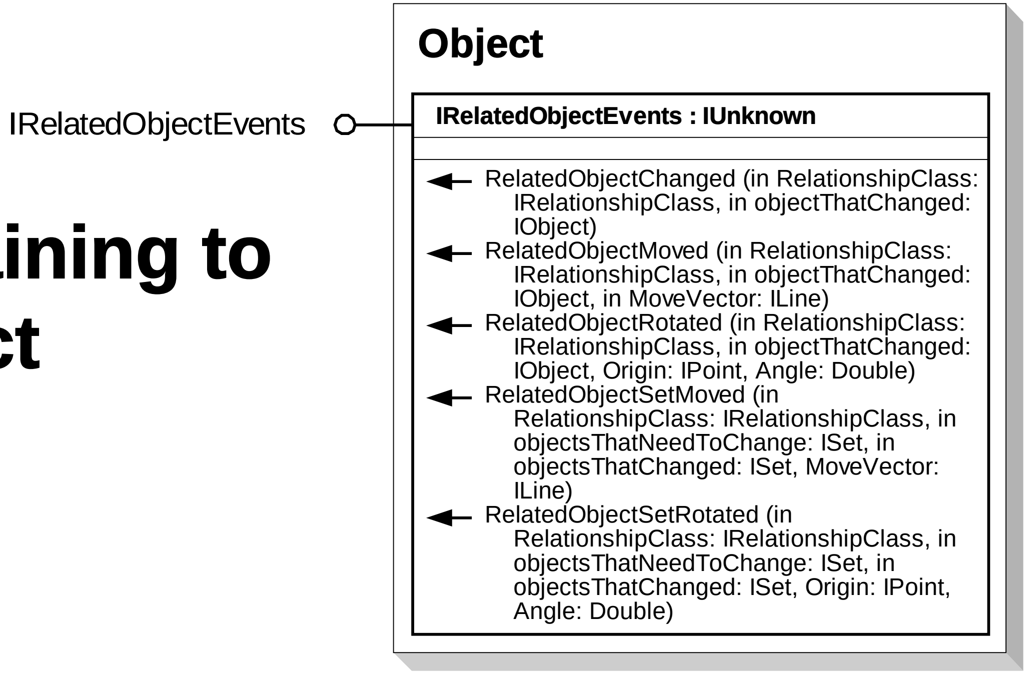
IRowChanges



IRowChanges
Row
IRowChanges : IUnknown
OriginalValue(in Index: Long): VARIANT
ValueChanged(in Index: Long): Boolean

- New with 8.1
- Useful for determining whether or not a field’s value has changed

IRelatedObjectEvents



IRelatedObjectEvents

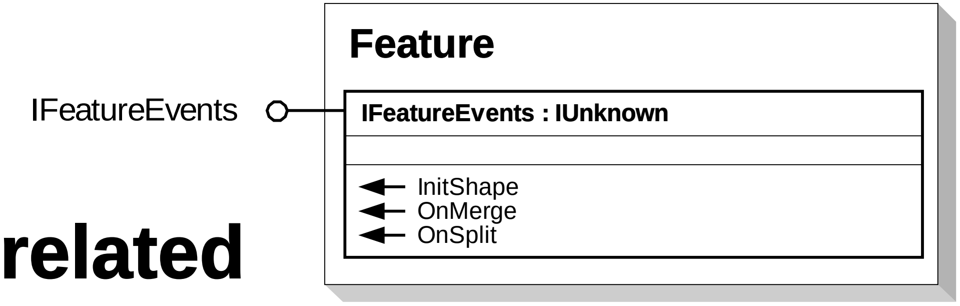
Object

IRelatedObjectEvents : IUnknown

- RelatedObjectChanged (in RelationshipClass: IRelationshipClass, in objectThatChanged: IObject)
- RelatedObjectMoved (in RelationshipClass: IRelationshipClass, in objectThatChanged: IObject, in MoveVector: ILine)
- RelatedObjectRotated (in RelationshipClass: IRelationshipClass, in objectThatChanged: IObject, Origin: IPoint, Angle: Double)
- RelatedObjectSetMoved (in RelationshipClass: IRelationshipClass, in objectsThatNeedToChange: ISet, in objectsThatChanged: ISet, MoveVector: ILine)
- RelatedObjectSetRotated (in RelationshipClass: IRelationshipClass, in objectsThatNeedToChange: ISet, in objectsThatChanged: ISet, Origin: IPoint, Angle: Double)

- **Events pertaining to related object modification**
  - **changing**
  - **rotating**
  - **moving**
- **Set-based methods for efficiency opportunities**

IFeatureEvents



Feature

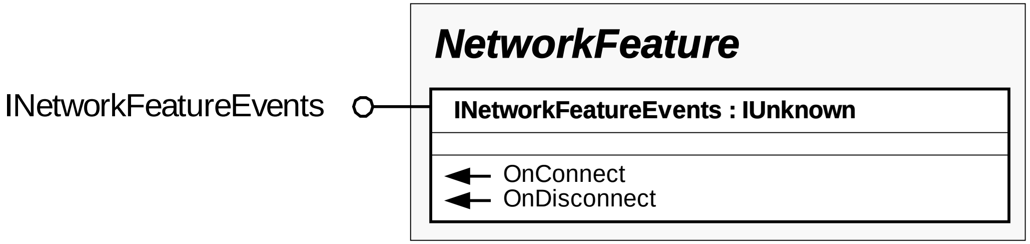
IFeatureEvents

IFeatureEvents : IUnknown

InitShape
OnMerge
OnSplit

- Events that are related to geometry changes
  - InitShape, OnMerge, OnSplit
- Good for
  - apportioning attributes in non-standard manners
  - initializing non-persisted connection points

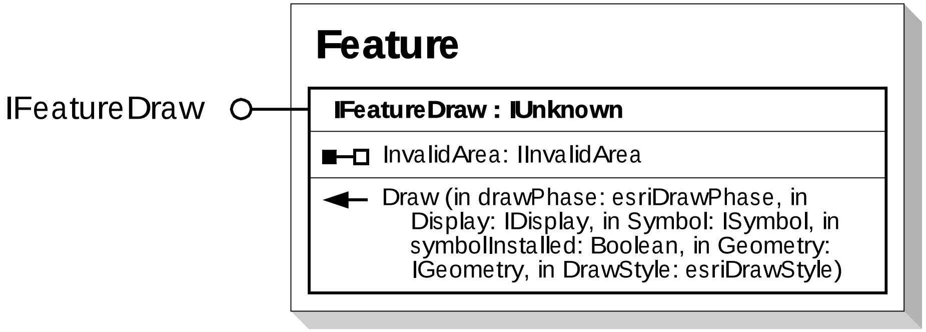
INetworkFeatureEvents



- Network connectivity events
  - OnConnect, OnDisconnect
- Unknown utility

STRIKE THIS?

IFeatureDraw



IFeatureDraw

Feature

IFeatureDraw : IUnknown

InvalidArea: IInvalidArea

Draw (in drawPhase: esriDrawPhase, in
Display: IDisplay, in Symbol: ISymbol, in
symbolInstalled: Boolean, in Geometry:
IGeometry, in DrawStyle: esriDrawStyle)

- **Used primarily with custom objects to override default drawing behavior**
  - **behavior that is inappropriate for a custom renderer**
- **Fairly simple to implement despite large argument list**

Common Navigation Tasks

Pseudo-C++ (ATL-based)

- Feature class

```
IObject::get_Class(IObjectClass**);
```

- Class extension

```
IObject::get_Class(IObjectClass** &ipClass);
ipClass->get_Extension(IUnknown** &ipUnk);
IClassExtensionPtr ipClassExtension(ipUnk);
```

- Feature dataset

```
IObject::get_Class(IObjectClass** &ipClass);
IFeatureClassPtr ipFeatClass(ipClass);
ipFeatClass->get_FeatureDataset(IFeatureDataset**);
```

Common Navigation Tasks

- **Geometric network**

```
IObject::get_Class(IObjectClass** &ipClass);
INetworkClassPtr ipNetworkClass(ipClass);
ipNetworkClass->get_GeometricNetwork(IGN** &ipGN);
```

- **Logical network**

```
IObject::get_Class(IObjectClass** &ipClass);
INetworkClassPtr ipNetworkClass(ipClass);
ipNetworkClass->get_GeometricNetwork(IGN** &ipGN);
ipGeometricNetwork->get_Network(INetwork** &ipNet);
```

Common Navigation Tasks

- **Related objects**

```
IObject::get_Class(IObjectClass** &ipOClass);
ipOClass->get_RelationshipClasses(relRole,
           IEnumRelationshipClass** &ipRelClasses);
IObjectPtr ipObject(this);
while (ipRelClasses->Next(&ipRelClass) == S_OK) {
  ipRelClass->get_ObjectsRelatedToObject(ipObject,
                  ISet** &ipObjects);
  while (ipObjects->Next(&ipRelatedObject) == S_OK){
    . . . whatever . . .
  }
}
```

Programming Caveats

- Do not assume too much – program defensively
  - check HRESULTs
  - assume your server components can fail
  - check arguments (inbound and outbound)
  - check for field existence
  - always obey the Rules of COM
- Consider the GDB versioning and transaction model during design
- Always minimize cursor creation

Programming Caveats

- Always bracket database edits inside an Edit Session (Start/StopEditing)
- Group changes inside of EditOperations (rollbacks)
- Always use NON-RECYCLING cursors when fetching data that will be updated

Programming Caveats

- Always retrieve all fields when searching for data that will be updated
- Always tag changed objects with store/delete to guarantee that object behavior is executed

Custom Object Demo

Conclusions

- Geodatabase provides large non-programmatic customization opportunities
  - most is built into core
- Creating custom objects requires
  - UML, CASE, VC++, COM
  - code generation and schema wizards help
- Time spent data modeling is very beneficial in the long run
- Pay attention to performance issues

For Further Info

For Further Info

- Relevant UC sessions:
  - *Overview of the Geodatabase*
  - *Designing and Using a Geodatabase*
  - *Working with a Versioned Geodatabase*
  - *Extending the Geodatabase with Class Extensions*
  - *Managing and Editing Geometric Networks*
  - *Geodatabase and Object Model Design Using CASE Tools*
  - *Working with Networks in ArcInfo 8*
  - *Advanced Customization with ArcObjects in C++*

For Further Info

- Geodatabase Literature
  - Michael Zeiler. *Modeling Our World: the ESRI Guide to Geodatabase Design*. ESRI Press, 1999.
  - Andy MacDonald. *Building a Geodatabase*. ESRI Press, 1999.
  - *Multi-user GIS Systems with ArcInfo 8*. ArcOnline White Paper, March 2000.

For Further Info

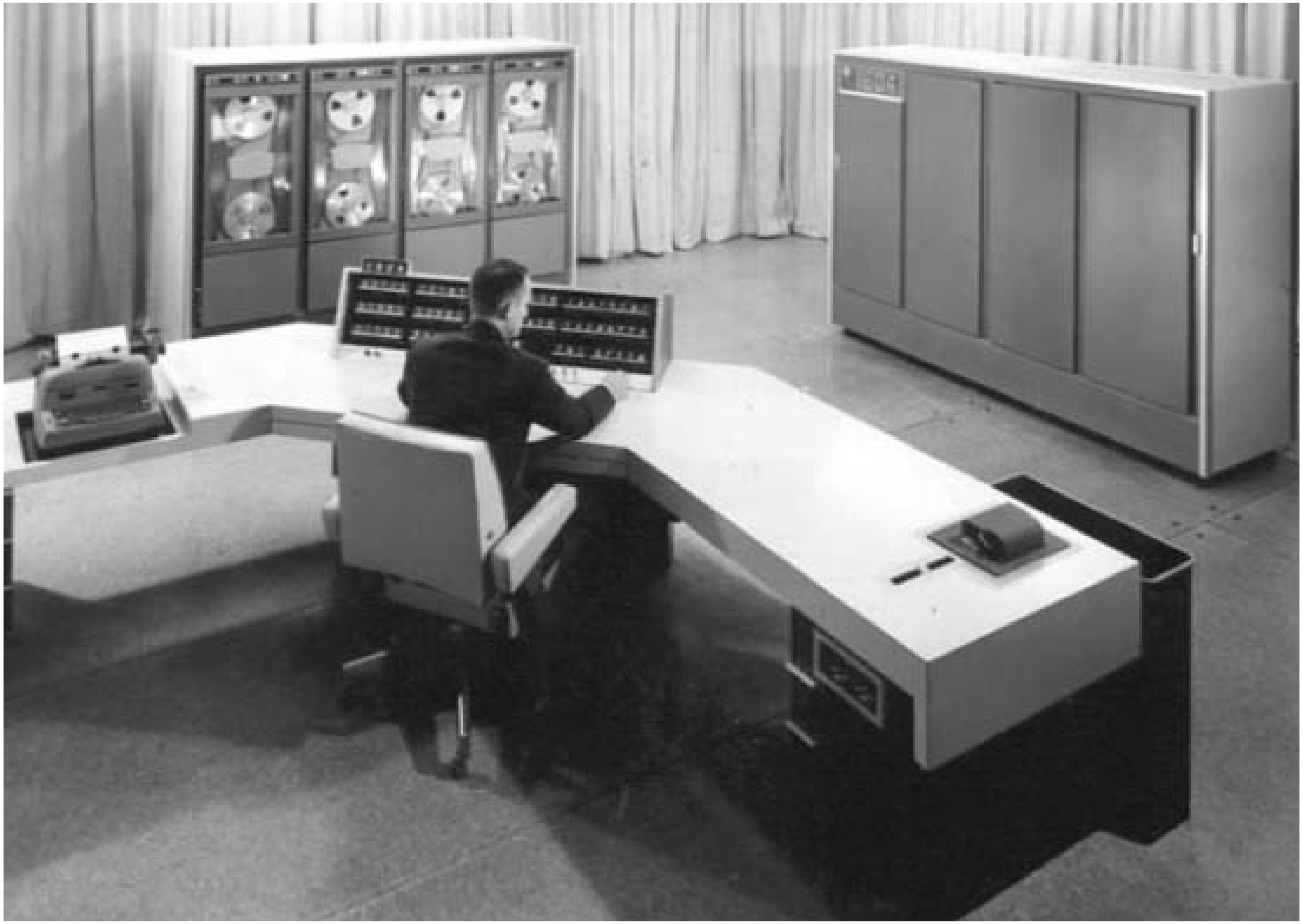
- General Literature
  - David Chappell. *Understanding ActiveX and OLE: A Guide for Developers and Managers*. Microsoft Press, 1996.
  - Dale Rogerson. *Inside COM: A Tedious Book for Superstar Geeks*. Microsoft Press, 1997.
  - Bertrand Meyer. *Object-Oriented Software Construction*. Prentice Hall, 1997.
  - Brent Rechtor, Chris Sells. ATL Internals, 1999.

geography
our global network
esri
©esri 2006

Programming Custom Objects with ArcInfo 8

Database Technology

- Many ways to model data
  - Graphic data model
  - Georelational data model
  - Object Relational d model



Startling Hi-Tech Demo