Extending the Geodatabase with Class Extensions

Erik Hoel

Brian Goldin

Goals

- Develop an understanding of
  - ArcInfo 8.1 Geodatabase
  - what customization opportunities are available
  - what class extensions are useful for
  - how to program class extensions
- How to proceed forward
  - other UC’00 sessions
  - literature

Agenda

- ArcInfo 8 Geodatabase
  - continuum of customization opportunities
- Class extensions
  - class extension interfaces
  - programming class extensions
  - managing class extensions
- Swank demo

ArcInfo 8 Geodatabase

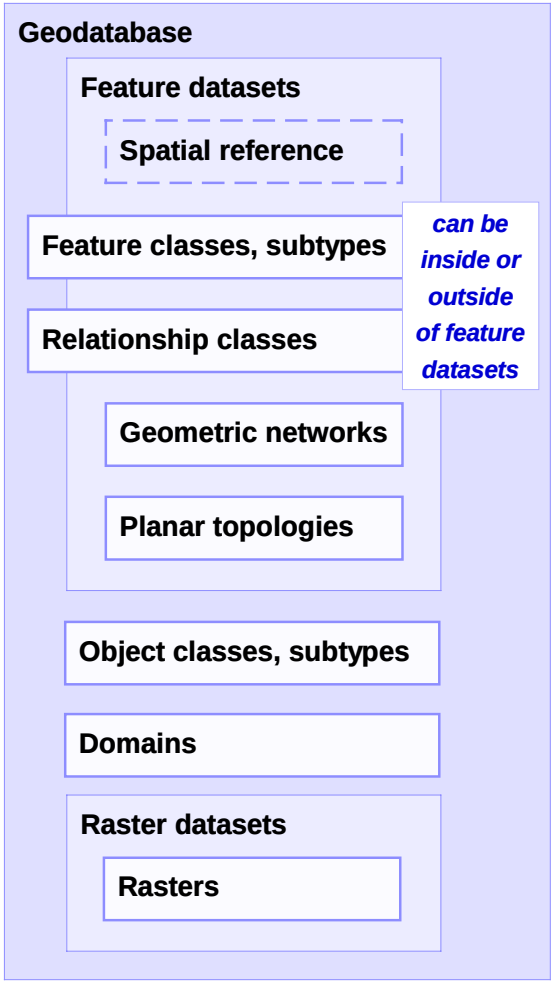
ArcInfo 8 Geodatabase

- 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

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

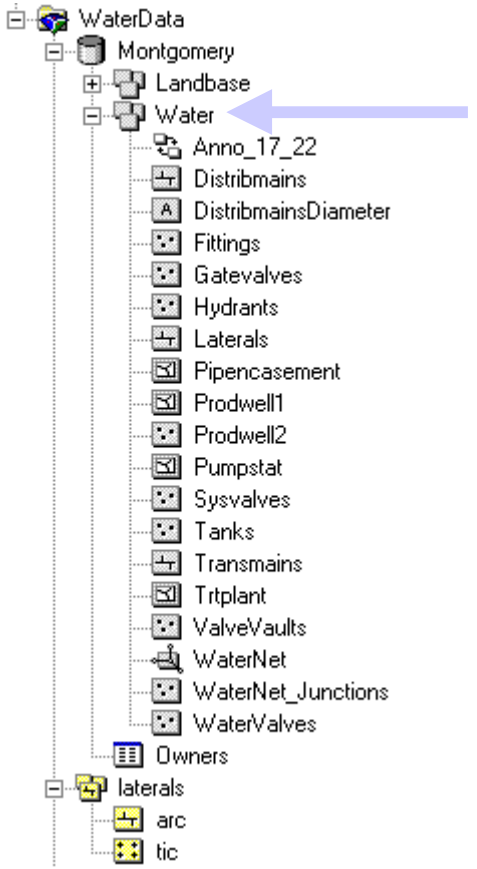
OID	Shape	Type	. . .
524	X,Y,Z,M, ...	Private	. . .

Feature (row)

- 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



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**

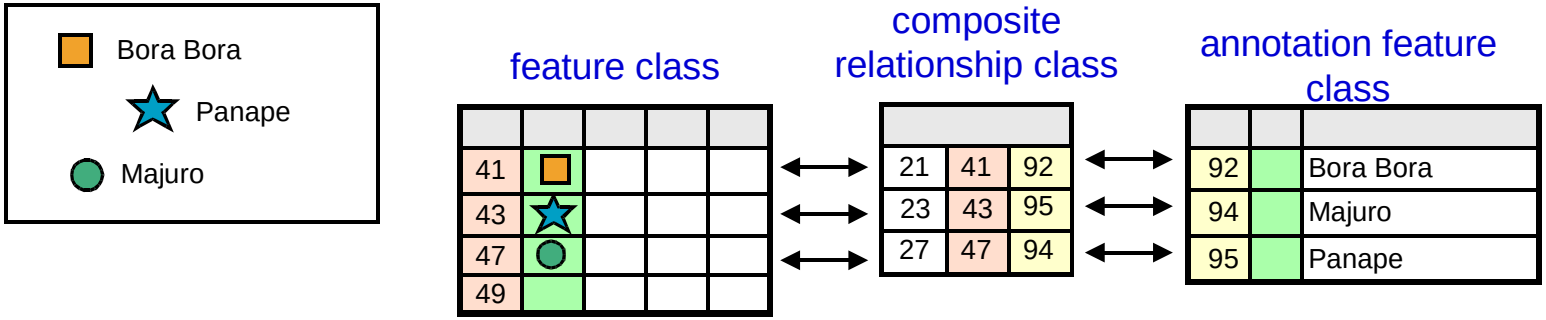
fid	geom	subtype	width	lanes	name
101		asphalt	85.3	4	Chimayo Highway
102		concrete	45.1	2	Acequia de Isabel
103		asphalt	75.9	4	Calle Petra
104		gravel	35.2	2	Maximilian Road

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…)

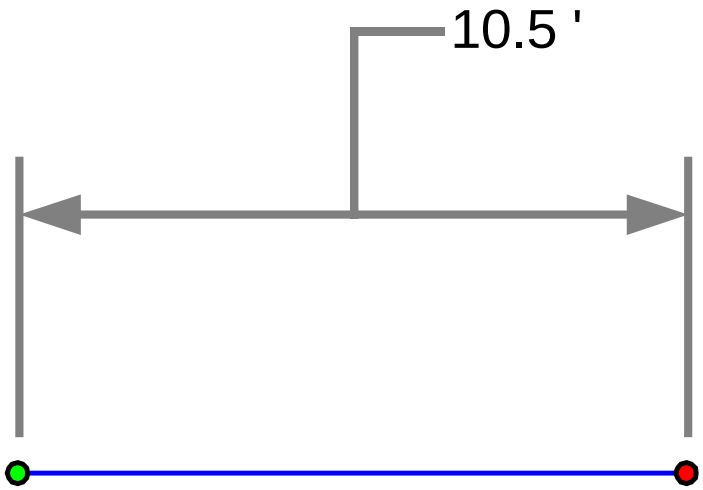
Annotation

- 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.



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

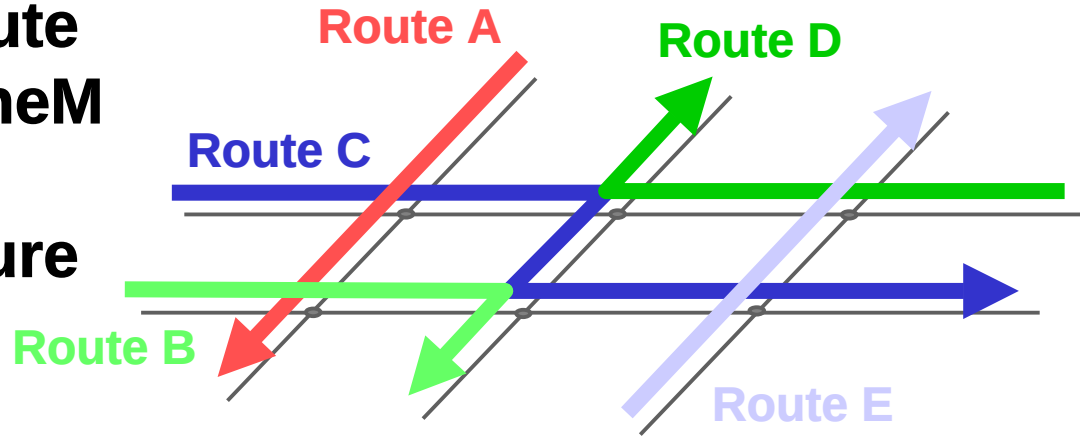
- Network features live in a geometric network
- Directly support network analysis
- Types:
  - simple junction
  - simple edge
  - complex junction
  - complex edge
- Integrity constraint:
  - edge must have a junction at each endpoint

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

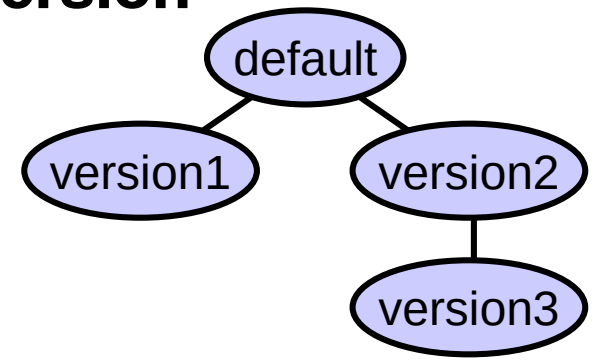


Planar Topology

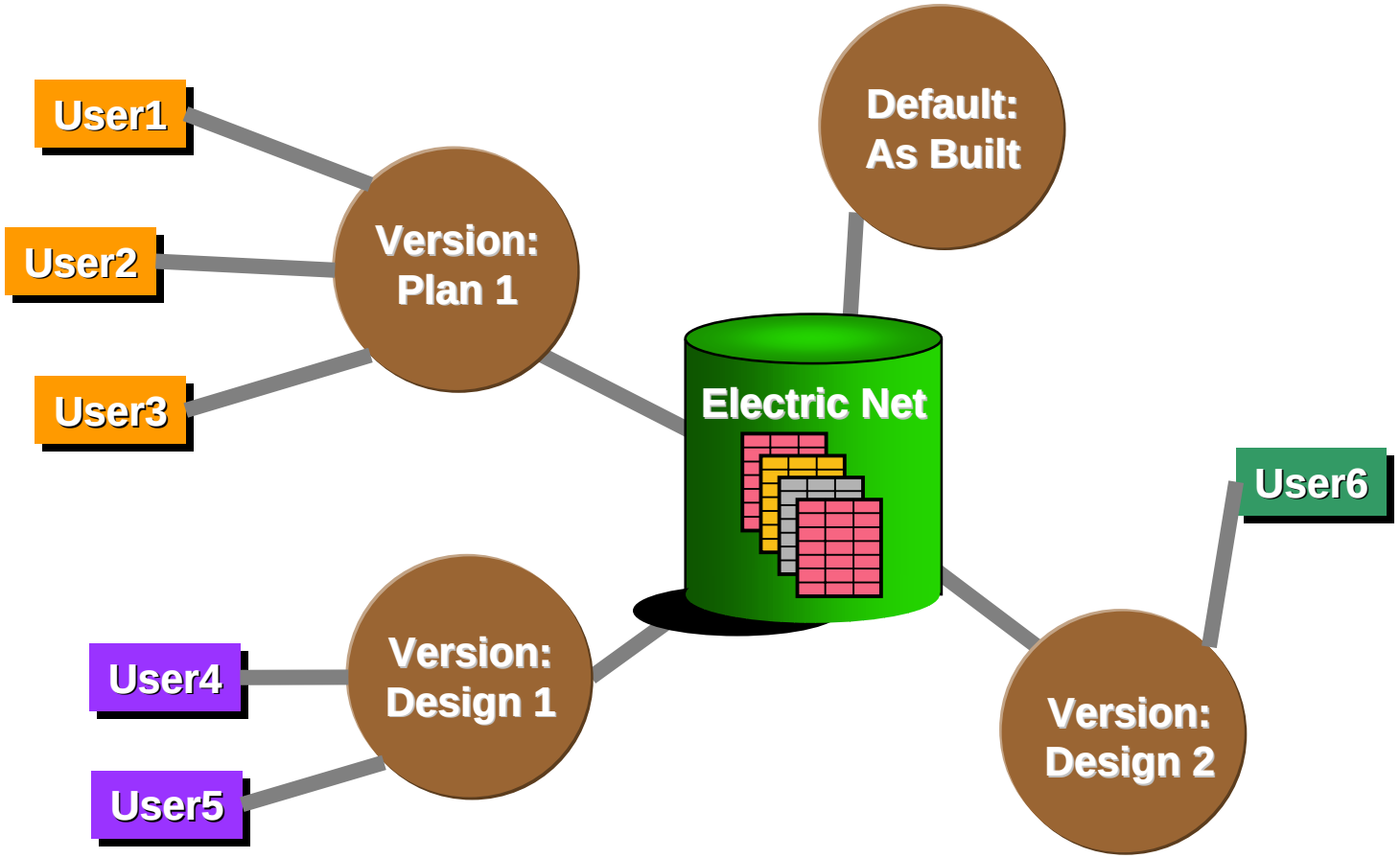
- Feature classes in an integrated feature dataset participate in a planar topology
- Features share boundaries
- Editor tools allow you to edit and maintain shared boundaries
- Use the *Integrate* command in the Editor to ensure coincident boundaries
- Use shared edge edit tool to edit shared boundaries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

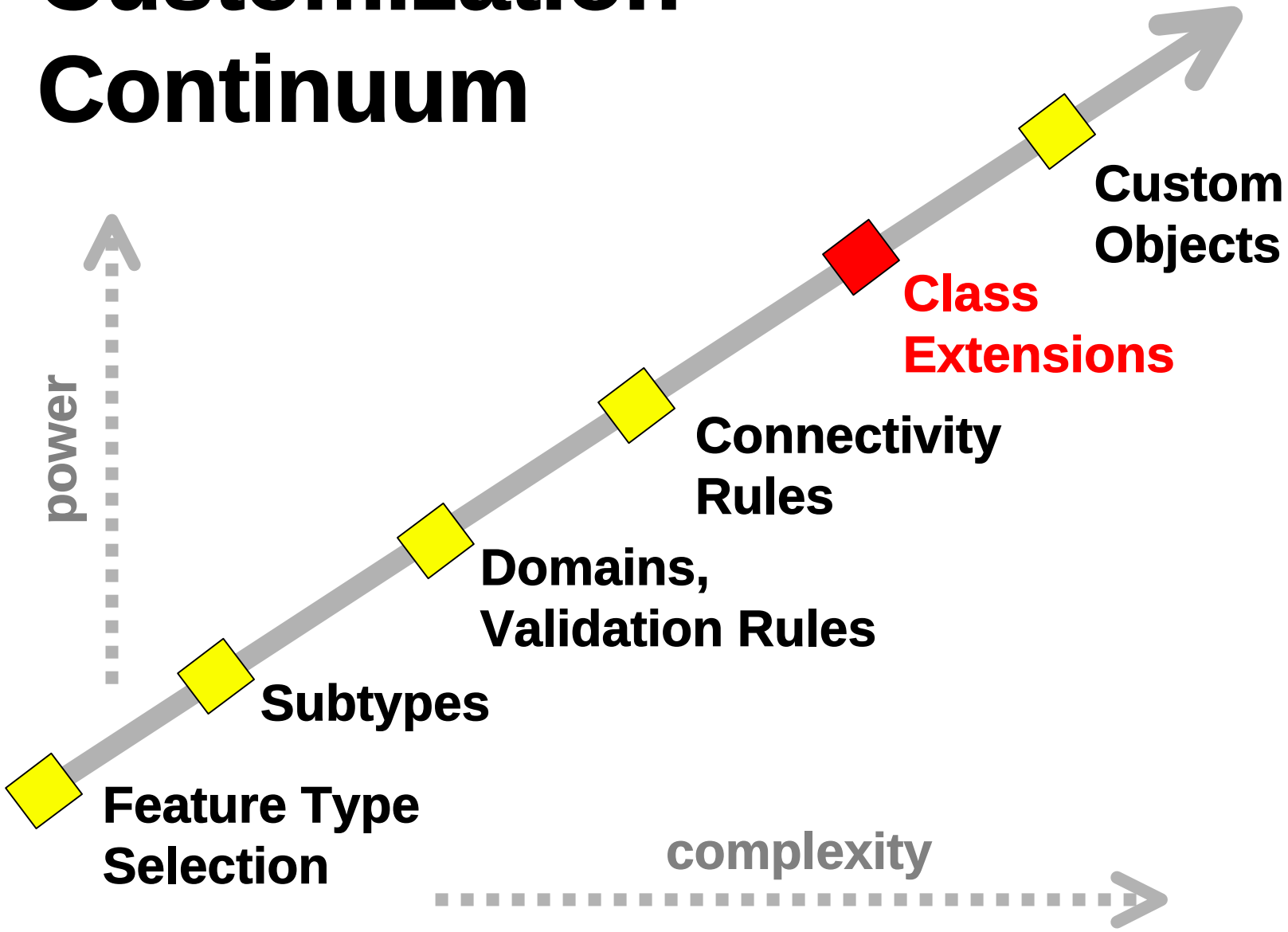


Multi-Versioned Database



Geodatabase Customization

Customization Continuum



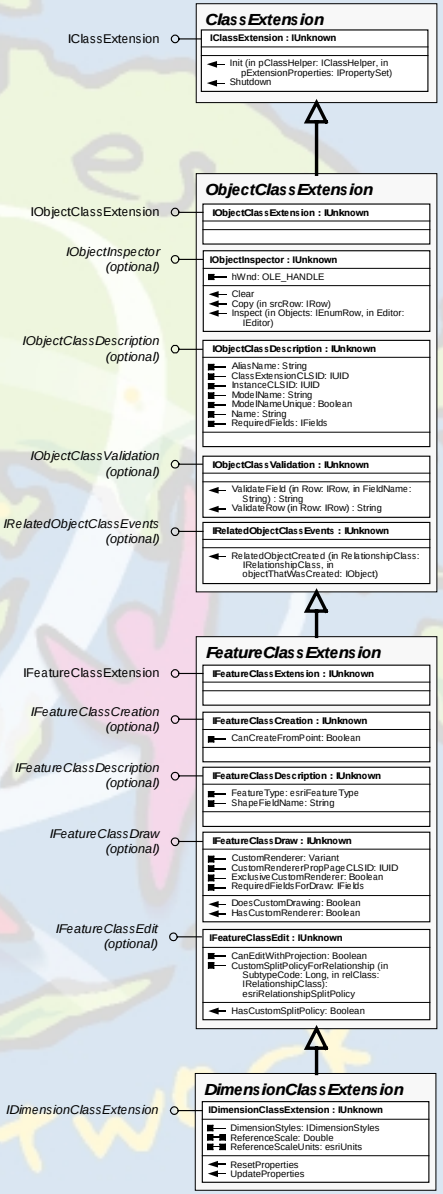
Customizing Existing Classes

- Define a new feature class (object class)
  - add attribute fields, set geometry type, spatial reference, etc.
- Import an object class
  - import template from another object class in any Geodatabase
- 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: C++, VB, Delphi
  - CASE tools and ESRI Code Generation and Schema Wizards make this a lot easier

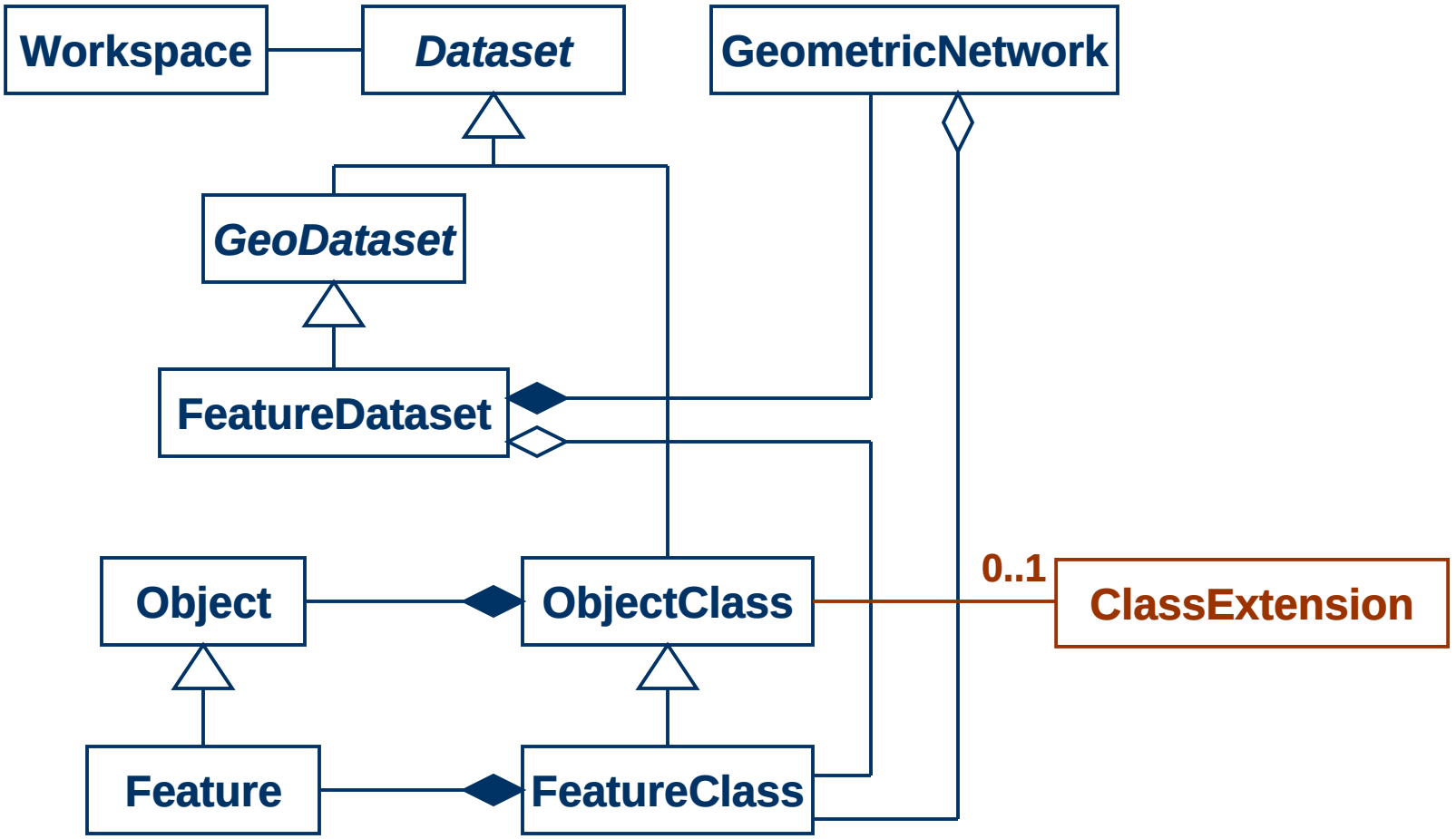
Class Extensions



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

Class Extensions



Class Extensions

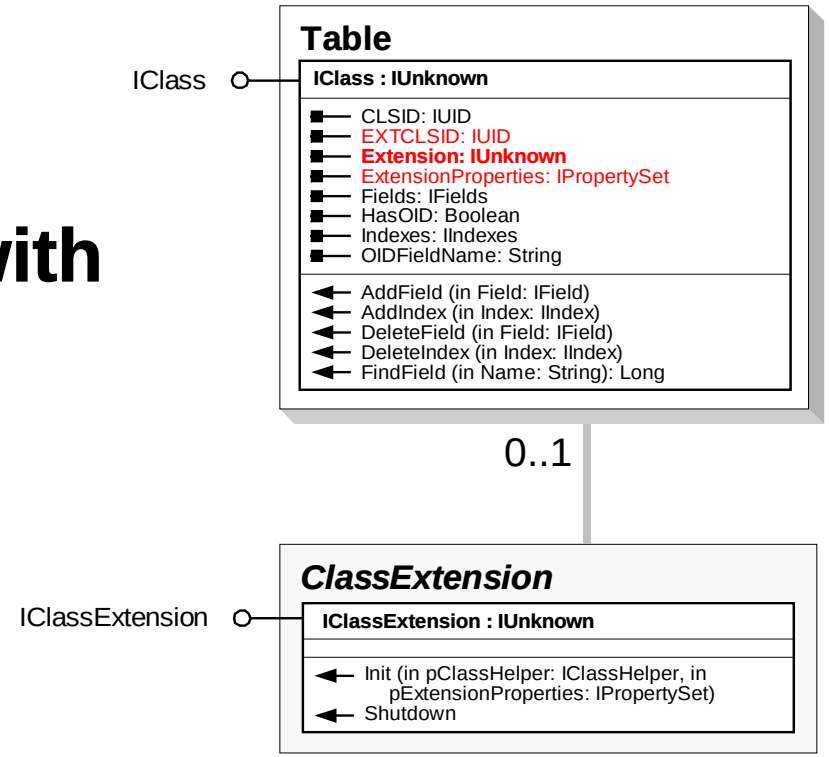
- Completely optional
  - Geodatabase tests when appropriate
- Implementable in VB (no aggregation)
- Large collection of specified interfaces
  - almost all are optional
  - Geodatabase tests for class extension and supported interfaces when appropriate
- One class extension per object class

Class Extension Interfaces

- One required interface
- Numerous other optional interfaces
  - schema generation
  - custom drawing
  - custom property inspection
  - custom validation
  - related object creation notification
  - custom split policies
  - anything else you specify for your client apps, tools, and features

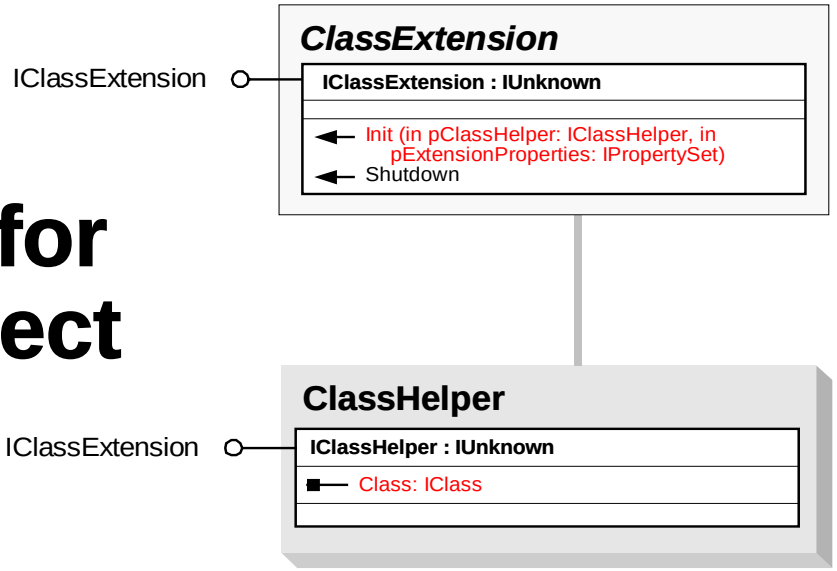
IClassExtension

- **This is the one required interface**
- **Init() always called with property set**
- **Shutdown() to kill circular reference**



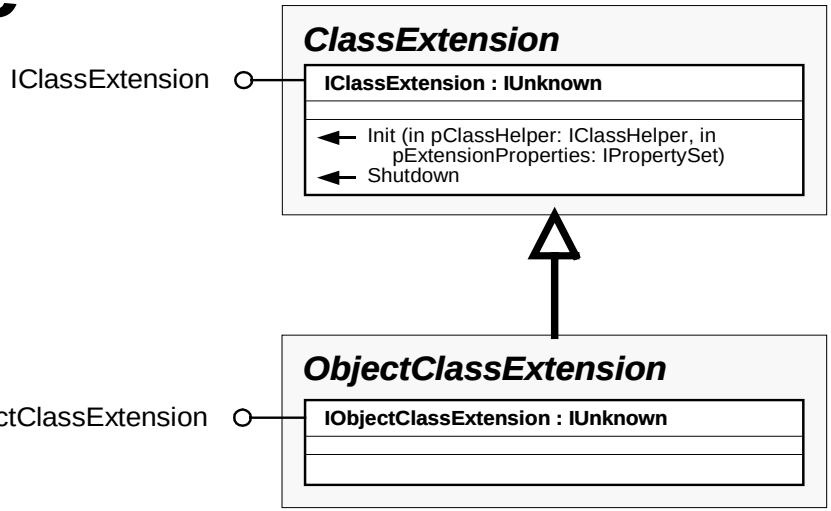
ClassHelper

- **Automatically populated helper for accessing the object class**
- **Argument to Init()**
- **Intended to prevent circular references**



IObjectClassExtension

- **Required interface for object class extensions**
  - **yes – I lied**
- **Empty interface**
- **Implementation not too difficult**

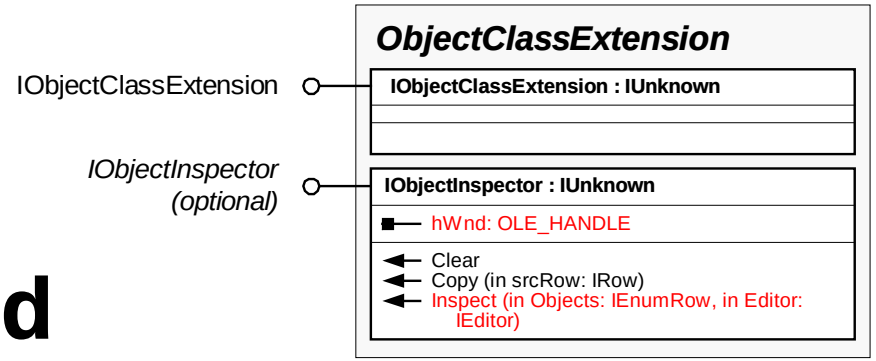


Extension Properties

- The ultimate hook
- Argument to Init()
- You can serialize anything you wish into the property set
  - class-level state information
  - image of your favorite record album
- Property set can be updated
  - update the current PropertySet
  - alter the class extension property set
- With power comes responsibility

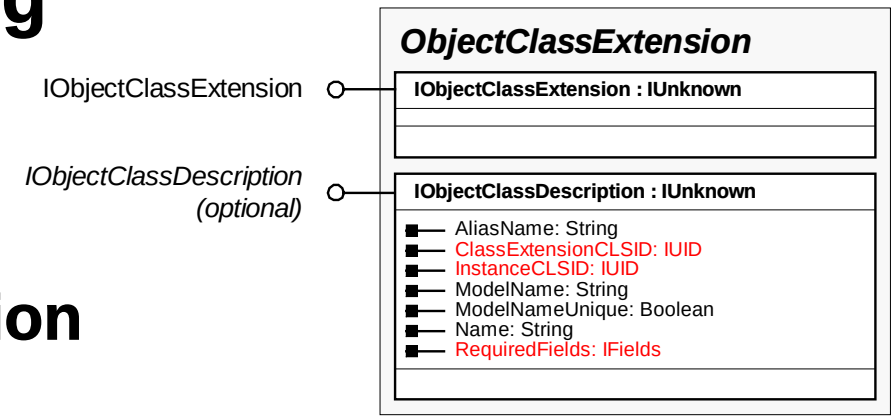
IObjectInspector

- **Optional**
- **Used for custom inspectors**
- **Can member in another VB-based inspector**



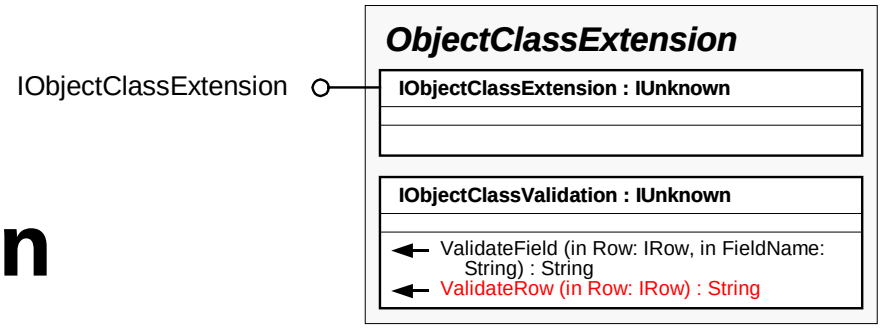
IObjectClassDescription

- **Optional**
- **Used by ArcCatalog to automate class creation**
  - **GUIDs for custom feature and extension**
  - **field set**
- **Unnecessary if using schema wizard**



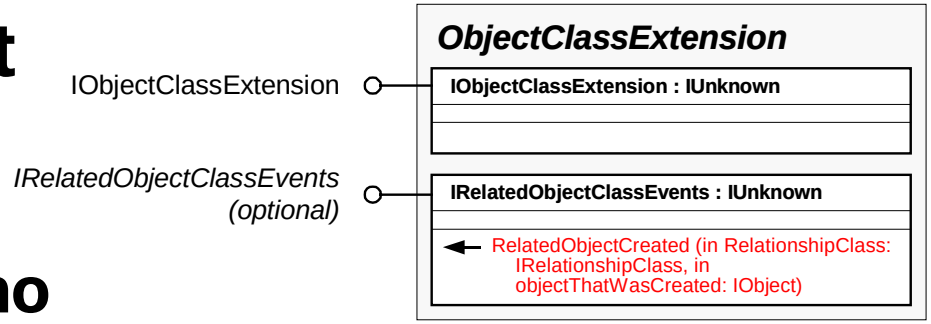
IObjectClassValidation

- **Optional**
- **Used to specify custom validation code**
- **Triggered during feature validation**



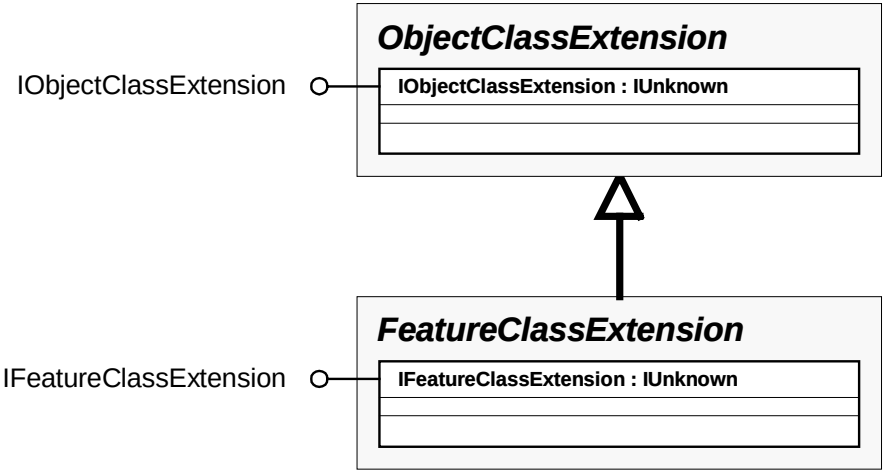
IRelatedObjectClassEvents

- **Optional**
- **Messaged when an object is created in a related object class**
  - **e.g., used by feature linked anno and dimension feature implementations**



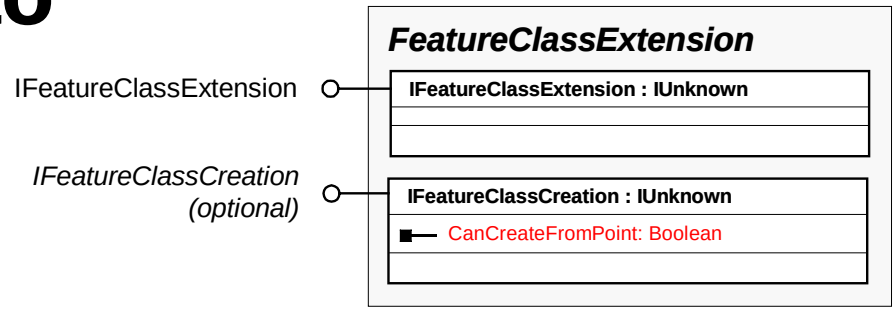
IFeatureClassExtension

- **Required interface for feature class extensions**
- **Empty interface**
- **Type info**



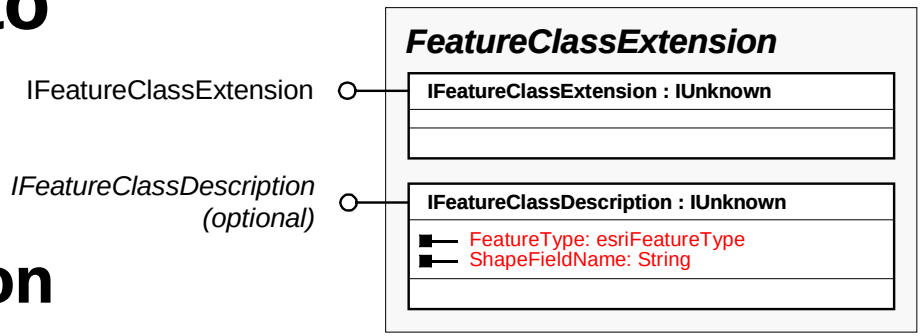
IFeatureClassCreation

- **Optional**
- **Used by Editor to know if feature can be created with single mouse click**



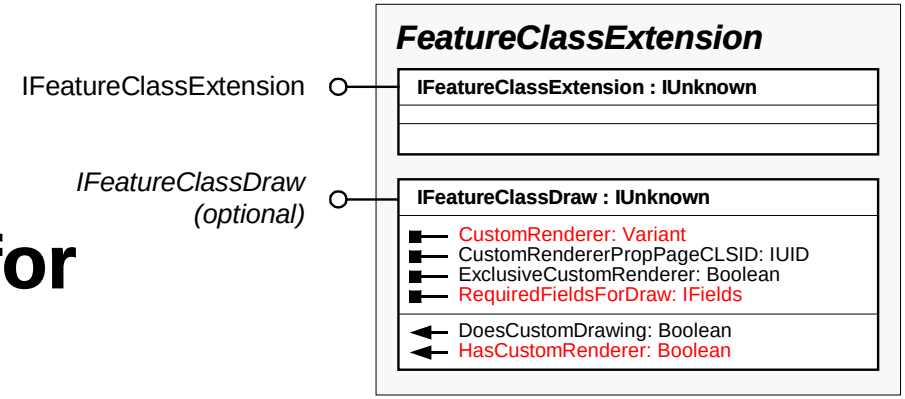
IFeatureClassDescription

- **Optional**
- **Used by Catalog to automate feature class creation**
  - **r.e., IOCDescription**
- **Unnecessary if using schema wizard**



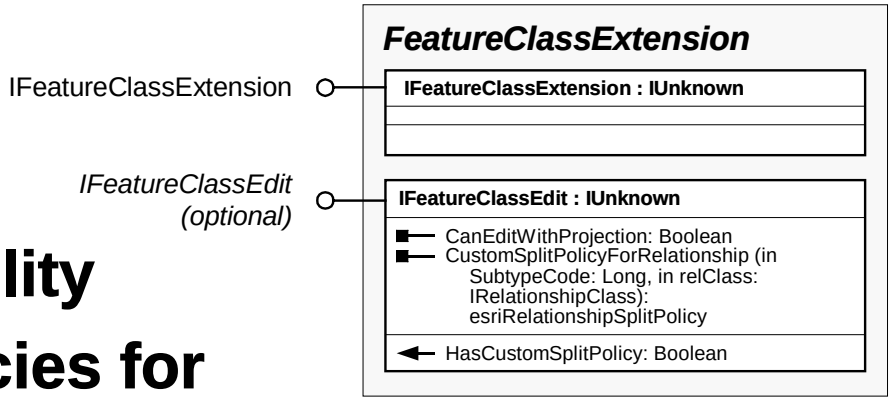
IFeatureClassDraw

- **Optional**
- **Used to specify**
  - **required fields for drawing**
  - **custom renderers**



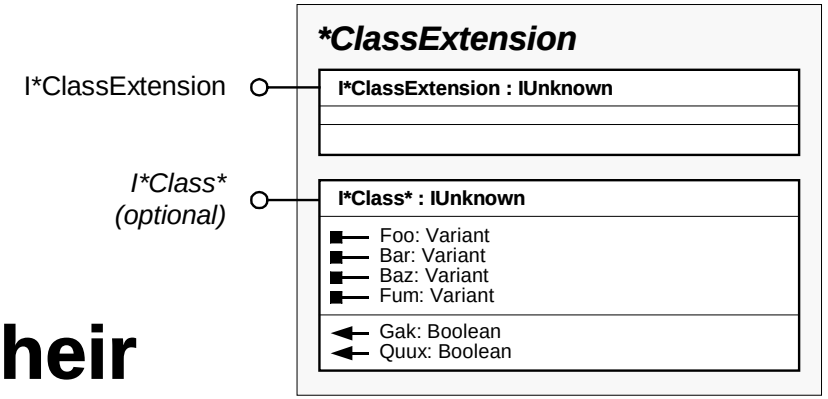
IFeatureClassEdit

- Optional
- New with 8.1
- Used to specify
  - projection editability
  - custom split policies for relationship handling
- Exposed at later time in ArcCatalog user interface



I*Class*

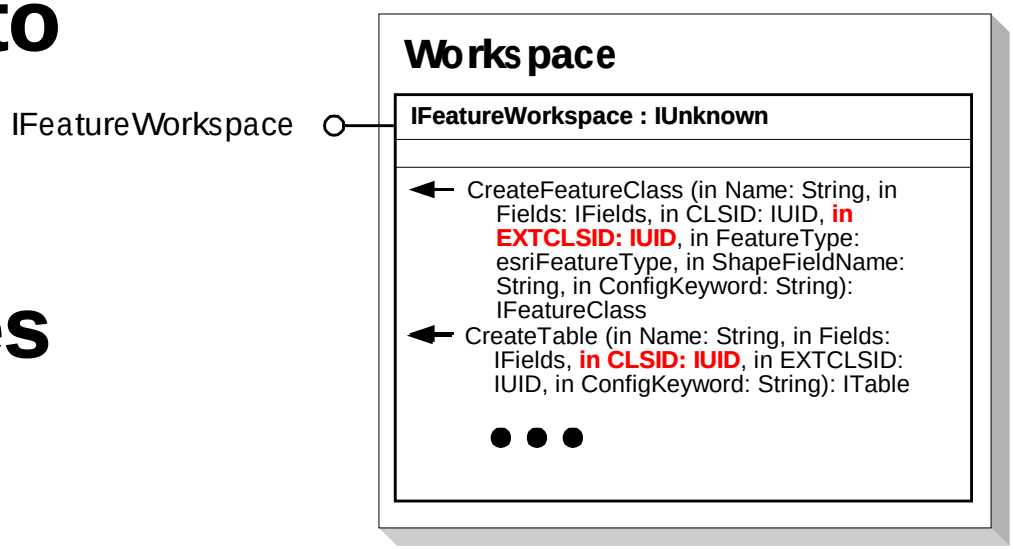
- **Additional new interfaces will be specified as GDB matures**
- **Users can specify their own**
  - **callable from their tools, apps, and custom features**



Managing Class Extensions

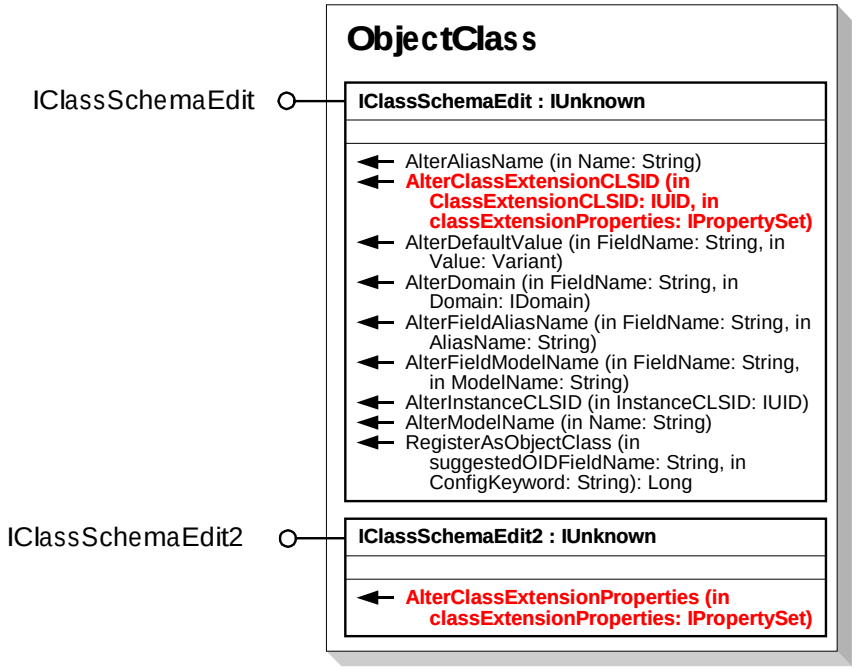
Managing Class Extensions

- **Assignment to new feature classes and object classes**



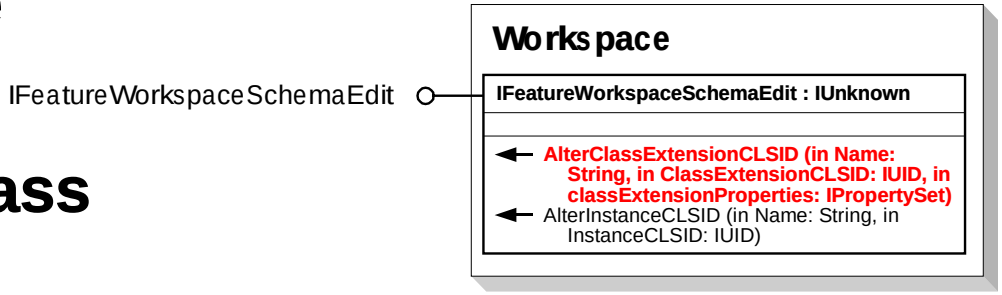
Managing Class Extensions

- **Updating class extension configuration**
  - **extension**
  - **properties**



Managing Class Extensions

- **Reconfiguration of class extensions on machines where the class cannot be opened**
  - **e.g., missing class extension**



Programming Class Extensions

Programming Class Extensions

- Class extensions requires programming in COM-compliant language
  - VB, VC++, VJ++, or Delphi
- CASE tools make it easier

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

Startling Demo

Conclusions

- Geodatabase provides large non-programmatic customization opportunities
  - most is already built in
- Creating class extensions requires
  - UML, CASE, VC++ or VB, COM
  - code generation and schema wizards help

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 Custom Objects***
  - ***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

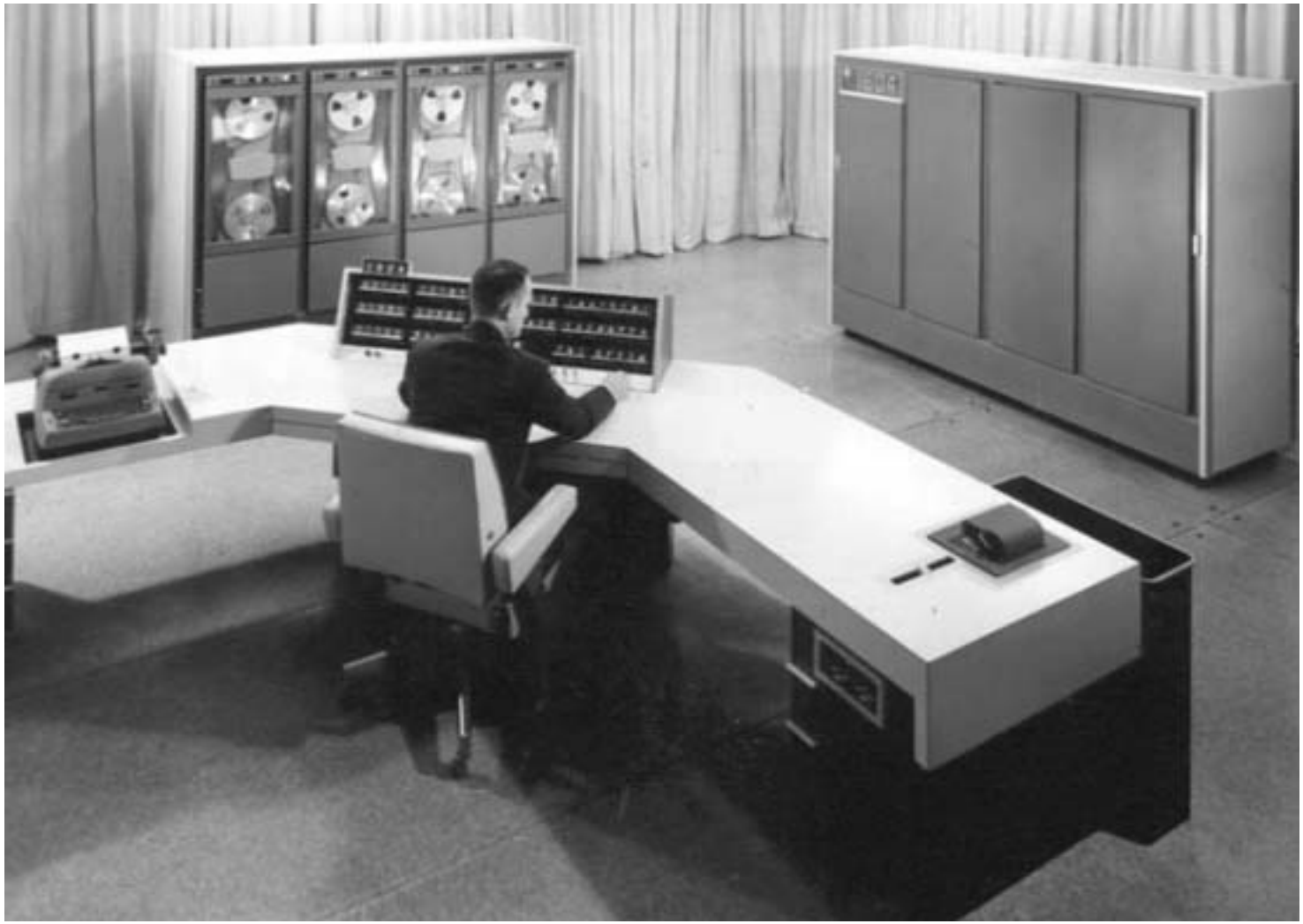
- **Geodatabase Literature**
  - **Michael Zeiler. *Modeling Our World: the ESRI Guide to Geodatabase Design*. ESRI Press, 1999.**
  - **Andy MacDonald. *Building a Geodatabase*. ESRI Press, 1999.**
  - ***Multi-user GIS Systems with ArcInfo 8*. ArcOnline White Paper, March 2000.**

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.**


geography
esri
gis
our global network

Programming Custom Objects with ArcInfo 8



Startling Hi-Tech Demo