Getting to Know ArcObjects

Rob Burke
ESRI-Redlands
Educational Services
• Started as an intern 1990 Tenth User Conference, PS
• Teach ESRI classes
  – Desktop, Geodatabase, ArcObjects, raster, Server
• Write training materials
• Write books
• Review books
Surveys

• Please fill out a survey
Intended audience

- Anyone wanting to learn about ArcObjects
- ArcObjects basics
- For programmers and non-programmers
- Maybe you know
  - Some ArcGIS
  - Some programming
- Not language specific
What are ArcObjects

Under The Hood

Diagrams

Interfaces & COM

Implement ESRI Interfaces

Extend the Applications

Extend the Geodatabase

CASE tools

Engine

Server

Wrap-up

Questions

Tech Talk Area

Lecture Path
Parts store for GIS

- Building blocks for ArcGIS software products and yours
- The most basic GIS parts are programmable objects
Parts store for GIS

- ArcMap and ArcCatalog are built with ArcObjects
- Their most basic parts are programmable GIS objects
- Users and programmers interact with the same objects
- Everyone uses the same parts, ESRI developers and you
Pieces to create GIS applications or embed

- Technically they are called classes (or components)
  - Over 3,400 ArcObjects classes
  - Over 21,000 properties and methods
  - Grouped into over 70 logical libraries
  - (not including ArcGIS Server)
- Each class corresponds to a basic GIS part
Map class vs map object

- There is only one Map class
- You make many map objects: World and US
- A class is code behind an object’s properties and methods
- Objects live in memory and take on your settings
How do you get ArcObjects

• ArcObjects is not a product, by itself

• You can’t buy just ArcObjects
  – You buy an ArcGIS Desktop product
  – You buy ArcGIS Engine
  – You buy ArcGIS Server

• ArcObjects-related files are installed when you install one of the three products
Use COM compatible languages

• VBA is built into the ArcGIS desktop applications
  - Visual Basic for Applications
  - VBA code is stored in map document files (mxd files)

• Buy a language and IDE (Development environment)
  - Visual Basic or C# (Visual Studio 2005)
  - C++
  - Java

• Use a free open source language and IDE
  - Python

  - My trail: BASIC, Fortran, SML, AML, Avenue, VB, VBA, Python, C#
Compare to Microsoft Access data objects

- Access built from a about a hundred objects
  - Called DAO or data access objects
- Users and programmers use same objects
- Applications can mix and match from different libraries
Class libraries

- COM classes can be mixed and matched
- Make your own library of COM classes
- Language independent

ArcObjects

MyClasses

Word

My Application
Shapefile to word doc

Word Doc

Feature Class
ArcObjects are building blocks: Desktop

- ArcGIS Desktop applications are created with ArcObjects
- ArcMap, ArcCatalog, ArcScene, ArcGlobe…
ArcObjects are building blocks: Engine

- ArcGIS Engine applications are created with ArcObjects
- You create stand alone GIS applications
- Use ArcObjects in non-GIS-centric applications
ArcObjects are building blocks: Server

• ArcGIS Server Web mapping applications are created with ArcObjects and special Web ADF GIS objects
• Author, publish, or consume
  – GIS data, functionality, and geoprocessing
  – Query, edit, address match, make custom, and more
ArcGIS: a complete GIS

Applications

- **ArcGIS Desktop**
  - Desktop Developer Kit
    - .NET
    - COM
  - Engine Developer Kit
    - .NET
    - COM
    - C++
    - Java

- **ArcGIS Explorer**
  - API
    - .NET

- **Web Mapping Application**
  - API
    - .NET
    - Java

- **ArcGIS Mobile**
  - SDK
    - .NET

Services

- ArcObjects

Data (Geodatabase)

- Personal for MS Access
- File
- Personal
- Workgroup
- Enterprise
- ArcSDE

ArcGIS Online

- ArcWeb Services
  - API
    - REST
    - J2ME
    - JavaScript
    - OpenLS
    - SOAP

Developer Summit 2007
Lecture Path

What are ArcObjects

Under The Hood

Diagrams

Interfaces & COM

Implement ESRI Interfaces

Extend the Applications

CASE tools

Extend the Geodatabase

Engine

Server

Wrap-up

Questions

Tech Talk Area

Developer Summit 2007 19
Object Model Diagrams

- Road maps to the ArcObjects classes
- Help you write code
- Based on UML – Unified Modeling Language
  - Symbols show relationships, connections, properties, and methods
- > 70 libraries
- > 110 posters
Where do you get the diagrams?

- Install the ArcObjects developer kit
  - C:\Program Files\ArcGIS\DeveloperKit\Diagrams\*PDF
- Help
  - Per library
- Online
  - EDN Web site
  - Per library
The twelve UML Symbols

- Relationships

- Classes

- Properties and methods
UML symbols

- Association
UML symbols

- Multiplicity
- Association
UML symbols

- Multiplicity
- Association

Diagram:
```
Farm --*-- Nest
  ^
  v
Chicken --*-- Nest
  |
  v
Wings
```

```
UML symbols

- Creates a
- Multiplicity
- Association

Diagram:
- Egg
- Chicken
- Nest
- Farm
- Wings
UML symbols

- Is composed of
- Creates a
- Multiplicity
- Association

Diagram:
- Egg
- Chicken
- Nest
- Farm
- Wings

Multiplicities: 1 for Egg, 2 for Wings
UML symbols

- Is a type of
- Is composed of
- Creates a
- Multiplicity
- Association

Diagram:

- Bird
- Farm
- Nest
- Chicken
- Egg
- Wings

Multiplicities:
- 2
- *
UML class symbols: Abstract class

- 2D and not shaded
- Objects can not be created from it
UML class symbols: Abstract class

- 2D and not shaded
- Objects can not be created from it
- Holds properties and methods that subclasses inherit
UML class symbols: CoClass

- 3D and shaded
- You create objects out of them
  - Declare a variable
  - Instantiate an object using the New keyword
UML class symbols: Class

- Other classes create or return these objects
- You write code with another object to create or get
  - You can’t create an egg without a chicken
  - You can’t get a wing without an chicken
UML property symbols

- Read and write property
  - These are attributes stored about the object
  - You can either get or set these properties

Chicken
- Age
- Color
- Name
UML property symbols

- Read only property
  - Left half barbell symbol
  - You can get this property’s value
  - But you can’t change it

Chicken
- Age
- Color
- Name
- Wing(side)
UML property symbols

- **Write only property**
  - Right half barbell symbol
  - Usually an edit property or like a password
  - You can change the value, but you can’t get it

Chicken

- Age
- Color
- Name
- Wing(side)
- Password
Property values

- Each property holds a value
- The values are of a certain type: Number, string, date, Boolean, object …
- The type appears to the right of the property name
  - Property name, a colon, and type
  - Name: String - means that the Name property holds a text string
  - The Wing property holds Wing objects
UML method symbol

- **Method**
  - Arrow symbol
  - Methods are actions the object can perform
  - Sometimes called behaviors

Chicken

← LayEgg
← Fly
Methods return values

- You write code to run a method
- Some methods return a value, some don’t
- The value’s type appears to the right of the method
  - Method, colon, and type of its return value
  - LayEgg: Egg - the LayEgg method returns an egg object
  - The Fly method returns nothing
Reading diagrams

- Classes are rectangles
- Classes have properties and methods
- Get neighboring or connected objects

```pLayer = pMap.Layer(0)```
ArcObjects classes are COM

• COM compliant
  – Component object model
  – Industry standard for creating classes
  – Programming language independent
  – COM classes can be reused between applications

• COM classes have programmer interfaces
  – Classes created in one language can communicate with other languages
  – ArcObjects classes are created in C++
  – Use them in C++, VB.NET, C#.NET, VBA, Python, etc.
COM classes have interfaces

- Objects are rectangles
- Interfaces (lollypops) group properties and methods
- Get neighboring or connected objects
  
  ```
  pLayer = pMap.Layer(0)
  ```
- Multiple interfaces

![Diagram of COM classes and interfaces](image)
Understanding data types

• Intrinsic data types
  – Numbers, strings, dates

• Simple objects (VB, Excel, MapObjects)
  – One default interface
  – Hidden

• ArcObjects
  – Multiple interfaces
Intrinsic data types and simple objects

'Intrinsic data
Dim x As Integer 'Declare
x = 4 'Set
MsgBox x * 10 'Use

'Simple Object
Dim b as CommandButton 'Declare
Set b = frmClock.cmdTime 'Set
b.Caption = "Time" 'Use

Integer
+ (add) - (subtract) * (multiply) / (divide)

CommandButton
Color
Enabled
Font
Caption
ToolTip

On a VBA form
Multiple interfaces

'ArcObjects
Dim p as IPoint    'Declare
Set p = New Point  'Set
p.zh = 5280        'Use

'Or
Dim p as IGeometry 'Declare
Set p = New Point  'Set
p.Projection = Albers 'Use
Client and server environment

- Client code instantiates a class
- Client only knows the methods exist but does not know how they are implemented on the server

```vbnet
Dim p as IPoint
Set p = New Point
p.z = 5280
```

Third-party developer code (e.g., VBA)
Dog COM class code

- **Interface module**
  - Define methods
  - No code

- **Class module**
  - Implement methods

ESRI’s ArcObjects code

Your code

- **Developer module**
  - Instantiate class
  - Call its methods
What COM interfaces provide

- Programming language independence
- The ability for functionality in applications to evolve over time
  - Add new interfaces without affecting client code
  - IDog, IDog2, IDog3, IDog4, IDog5
  - Dim d as IDog works forever
- Interface reuse
- Your classes can implement ESRI interfaces!
Implement ESRI interfaces

- **Interface module**
  - ESRI’s ArcObjects code
  - Your code

- **Class module**
  - Implement methods
  - Write your own code

- **Developer module**
  - Instantiate class
  - Call its methods

```vbscript
' Instantiate Dog
Set Rover = New IDog

' Call Dog bark method
Rover.Bark
```

```vbscript
Public Sub Bark()
    ' Barking method
    MsgBox "Grr, Roof!"
End Sub
```

```vbscript
Public Sub IDog:Bark()
    MsgBox "Grr, Roof!"
End Sub
```

```vbscript
Sub Dog:Bark()
    Dim Rover As IDog
    ' Instantiate Dog
    Set Rover = New Dog
    ' Call Dog bark method
    Rover.Bark
End Sub
```

```vbscript
Public Sub IDog:Bark()
    MsgBox "Grr, Roof!"
End Sub
```

```vbscript
Sub Dog:Bark()
    Dim Rover As IDog
    ' Instantiate Dog
    Set Rover = New Dog
    ' Call Dog bark method
    Rover.Bark
End Sub
```
Creating a custom command

- Single-click buttons
- Execute any ArcObjects code
- Examples
  - Custom add data
  - Custom mapping operations
  - Show a custom dialog or window

ESRI Add Data command

My custom command
Developing custom COM components

• Find a similar ArcGIS component
• Find out what interfaces it implements
  – Look in the Help
• Buttons implement ICommand
Commands: Implement the ICommand interface

- Model after the Button class
- You get to code each property and method
- ESRI developers follow this pattern
- You can too!
Developing custom COM components

- Create a COM object and plug it into an application
- Steps:
  1. Create a COM/.NET project
Developing custom COM components

- Create a COM object and plug it into an application

Steps:
1. Create a COM/.NET project
2. Create a COM class
Developing custom COM components

- Create a COM object and plug it into an application
- Steps:
  1. Create a COM/.NET project
  2. Create a COM class
  3. Reference the ArcGIS libraries
  4. Implement ArcObjects interfaces
Developing custom COM components

- Create a COM object and plug it into an application
- Steps:
  1. Create a COM/.NET project
  2. Create a COM class
  3. Reference the ArcGIS libraries
  4. Implement ArcObjects interfaces
  5. Compile
  6. Register in an ArcGIS component category (Next Slide)
How does ArcMap know to use my class?

- ArcMap application start up cycle starts
  1. Accesses the appropriate component category
Application start up cycle

- ArcMap starts
  1. Accesses the appropriate component category
  2. Creates an internal object that implements ICommand

Diagram:

1. Application Starts

2. Command Created

- ESRI Component Categories
- ESRI Mx Commands
- ESRI Mx CommandBars
- ICommandItem
- Cmd
Application start up cycle

- ArcMap starts

1. Accesses the appropriate component category
2. Creates an internal objects that implements ICommand
3. Creates your UI component (command, tool, toolbar, or menu)
Application start up cycle

- **ArcMap starts**
  1. Accesses the appropriate component category
  2. Creates an internal objects that implements ICommand
  3. Creates your UI component (command, tool, toolbar, or menu)
  4. Adds the CommandItem to the CommandItem list
Custom tools also implement ITool

- Commands that listen for mouse and key events
- Allows users to interact with maps
- Examples
  - Map interaction: Tracking
  - Analysis: Select features
  - Editing tools
Custom toolbars

- Container to show commands, tools, and menus
- Implement IToolbarDef
- You code the interface’s properties and methods
- Examples
  - Show or hide a custom set of tools
  - Associate tools with extensions
ArcGIS extensions

- Mechanism to plug objects into the application
- Visible in the Extension Manager
- Supported by all applications
- Name appears in the list
- Looks like ESRI’s extensions

- Description appears at the bottom
  - Extension name and version
  - Copyright and company name
  - Extension purpose
Extension interfaces

• IExtensionConfig is required
  – Can be like a silent, invisible extension, and won’t appear in list
  – Load data
  – Check status

• IExtension is optional
  – Adds extension to the Extension Manager window’s list
  – Could load your toolbar and commands
Add table of contents tabs

• Implement IContentsView
Add ArcCatalog tabs

- Implement IGxView
API objects match your experience as a user

- Programmers start the same place users start
API objects match your experience as a user

- Programmers start the same place users start
- Geodatabase API uses the same (user) terms
- Includes any data, not just geodatabase formats
Some Geodatabase API objects

- **ObjectClass** is a regular table
- **FeatureClass** is a table with shape field

![Diagram of Geodatabase API objects](image-url)
Levels of customization

- Custom features
- Class & workspace extensions VB & C# >
- Editor VBA>
- Connectivity & topo rules
- Domains & validation
- Subtypes & defaults
- Built-in Functionality
- Custom Applications

Developer Summit 2007
Levels of GeoDatabase customization

- **Clients**
  - ArcMap, ArcCatalog
  - Custom Application

- **Geodatabase Class Extensions**
Level of customization

• Application level
  – Business Logic is stored within application
  – Example, a new button, a new edit task
  – Problem - duplication in many applications

• Database level
  – Business Logic is stored with data
  – Always available, regardless of application
  – Problems, messages are firing, row/class behavior, and code failure renders data useless
Extendable objects

- WorkspaceExtension
  - Workspace
  - Dataset
    - GeoDataset
    - FeatureDataset
    - ObjectClass
    - FeatureClass
  - ClassExtension
    - FeatureClass Extension
Class extension facts

• They are *not* software extensions that you load
  – Like Spatial Analyst

• You add a layer to a map, you get the behavior

• Behavior *lives* in the database

• One class extension per class
Class extension uses

- Schema generation
- Custom drawing
- **Custom property inspection** (next)
- Validation
- Custom split policies
- Related object creation notification
Class extension example

- ESRI’s Attribute inspector
- From the Editor toolbar
Custom inspector

- Extend behavior of an entire feature class
- Your window appears instead of ESRI’s
- Behavior stored with the data, not in an mxd file
Make your own custom feature classes

- IFeatureClassDraw
  - Override a feature class’s drawing for any client
  - You can make a custom renderer and property page for it
- Control new class creation
  - User can create your feature class

Your description here
(In ArcCatalog)
Behavior interfaces: Listen and React

- IObjectClassValidation

Validation event:
Building height must be 10 times the number of stories
Behavior interfaces: Listen and React

- IObjectClassValidation
- IRelatedObjectClassEvents
- IConfimSendRelatedObjectEvents

Validation event:
Building height must be 10 times the number of stories

As a building is added, notify related parcel to update its structure count
Behavior interfaces: Listen and React

- IObjectClassValidation
- IRelatedObjectClassEvents
- IConfimSendRelatedObjectEvents
- IObjectClassEvents \((OnCreate, OnDelete, OnChange)\)

Validation event:
Building height must be 10 times the number of stories

OnModify:
Record current time and user name in the table

As a building is added, notify related parcel to update its structure count
Registering class extensions with the geodatabase

- Limit of one class extension per class
- Class extension’s GUID must be registered on all client machines accessing data
- Methods to register extension:
  - During creation: IFeatureWorkspace CreateFeatureClass
  - After creation: IClassSchemaEdit AlterClassExtensionCLSID
  - When modeling and designing in Visio

The object classes table within a geodatabase

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>CLSID</th>
<th>EXTCLSID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Parcels</td>
<td><code>{3070721...</code></td>
<td><code>{0368CF51...</code></td>
</tr>
<tr>
<td>2</td>
<td>Buildings</td>
<td><code>{3070721...</code></td>
<td><code>{044782D...</code></td>
</tr>
<tr>
<td>3</td>
<td>Owners</td>
<td><code>{3070721...</code></td>
<td><code>{044782D...</code></td>
</tr>
</tbody>
</table>
CASE Process

- UML ➔ Repository ➔ GDB schema or C++ code
Visio application

Stencils

Drawing page
Extending ESRI classes

- Subclasses of *Feature*
- Add attributes, relationships, or subtypes
- *Attributes* become fields in the table

```plaintext
Feature
+Shape : esriFieldTypeGeometry

MyParcel
-ParcelValue : esriFieldTypeInteger
-CombinedBuildingValue : esriFieldTypeInteger

MyBuilding
-Stories : esriFieldTypeInteger
-Height : esriFieldTypeInteger
-BuildingValue : esriFieldTypeInteger
-ParcelID : esriFieldTypeInteger
```
Using the ArcCatalog Schema Wizard

- Select a geodatabase in ArcCatalog
- Click the Schema Wizard button
- Choose a repository

Select the XMI file.

Model stored in XMI file
Model stored in Repository database

Database Path or Connection String:
C:\{DPET\TestModel\manhattanKS.xml

User Name:
Password:

< Back Next > Cancel

Repository
Create schema

Visio
ArcGIS Engine

- Standalone
- Non-visual
- Embedded applications
ArcGIS Engine

Two Products

• Engine Developer Kit is the toolkit for building custom GIS and mapping applications

• Engine Runtime is deployable ArcObjects required to run custom Engine applications
Developer Controls

- MapControl
- PageLayoutControl
- ToolbarControl
- TOCControl
- ReaderControl
- SceneControl
- GlobeControl
The Map Control

- Put the map control in a Word document or your application
Tools and Commands

More than 100 tools and commands included
ArcGIS Engine Functionality

- Read all supported ESRI data formats including the geodatabase
- Map authoring (create and edit MXD)
- ArcGIS level cartography
- Query and analysis
- Geocoding
- Simple editing (shp and pGDB)

“ArcView without the applications”
Learning ArcGIS Server

• Training classes
  – Intro – Two day class
  – Developing – Three day class
• EDN with walk-throughs
• Forum discussions
• Blog with examples
Lecture Path

What are ArcObjects

Under The Hood

Diagrams

Interfaces & COM

Implement ESRI Interfaces

Extend the Applications

CASE tools

Engine

Extend the Geodatabase

Tech Talk Area

Wrap-up

Questions

Server

Developer Summit 2007
Book sales at the Spatial Outlet

• The Spatial Outlet is in Mesquite rooms G & H
• The hours are:
  – Tues: 9-6
  – Wed: 9-6
• All books: 50% off list price
Surveys

• Please fill out a survey
Further questions?

• Our TECH-TALK AREA
  – Where: TECH-TALK area 5
  – When: during the next 30 minutes
  – What: Opportunity to discuss questions and concerns