2008 ESRI Developer Summit

March 17–20, 2008 • Palm Springs, CA

Extending ArcGIS Desktop Applications with .NET

Katy Dalton

Kevin Deege

Schedule

- 75 minute session
  - 60 – 65 minute presentation
  - 10 – 15 minutes Q & A following the presentation
- Cell phones and pagers

- Please complete the session survey – we take your feedback very seriously!

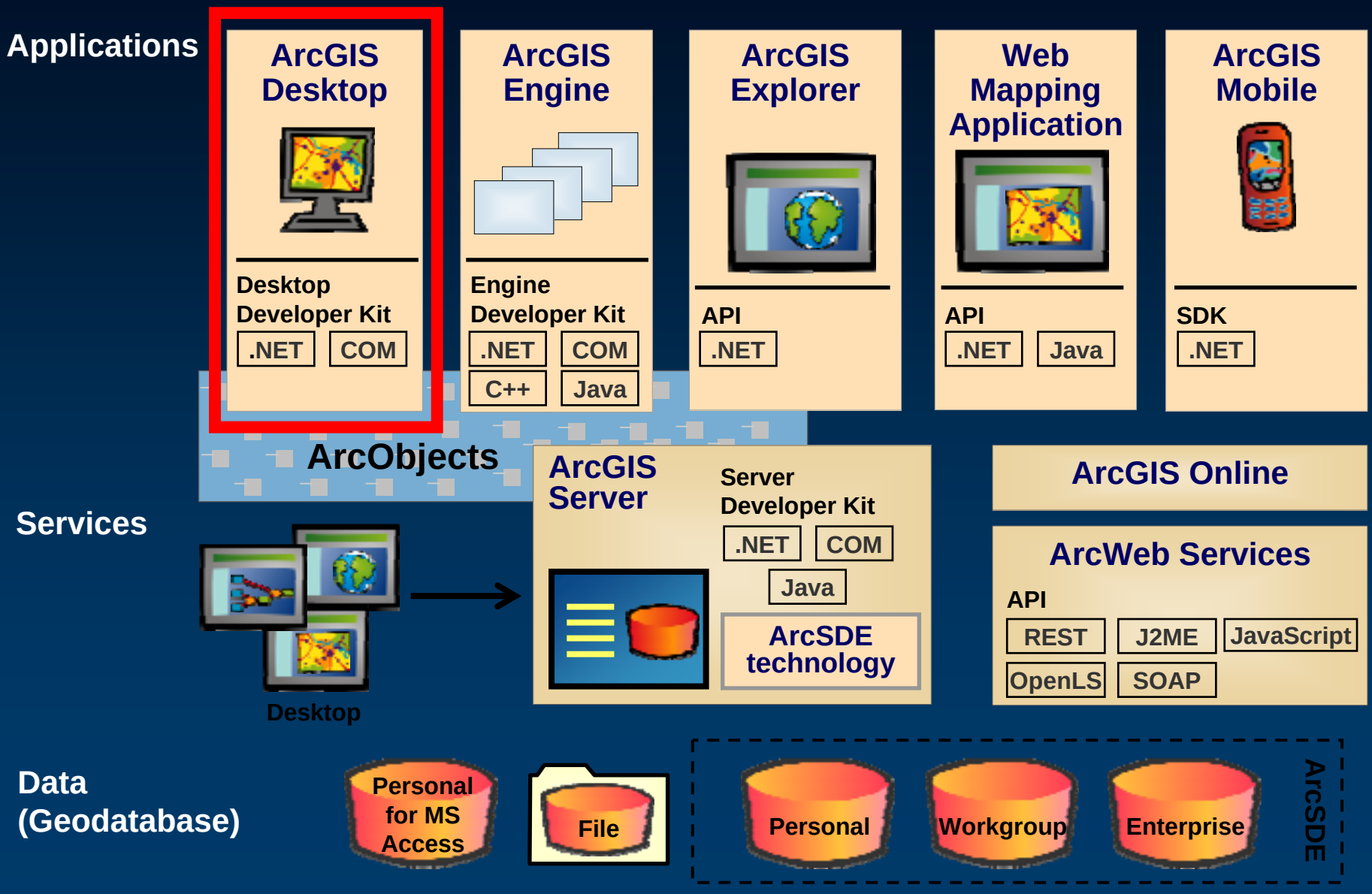
Introductions

- **Who we are:**
  - **Katy Dalton – ArcGIS .NET SDK team**
  - **Kevin Deege – ArcGIS ArcObjects Instructor**
- **Who you are:**
  - **You are a developer – comfortable with some kind of .NET programming**
  - **You are familiar with ArcObjects.**

Presentation Outline

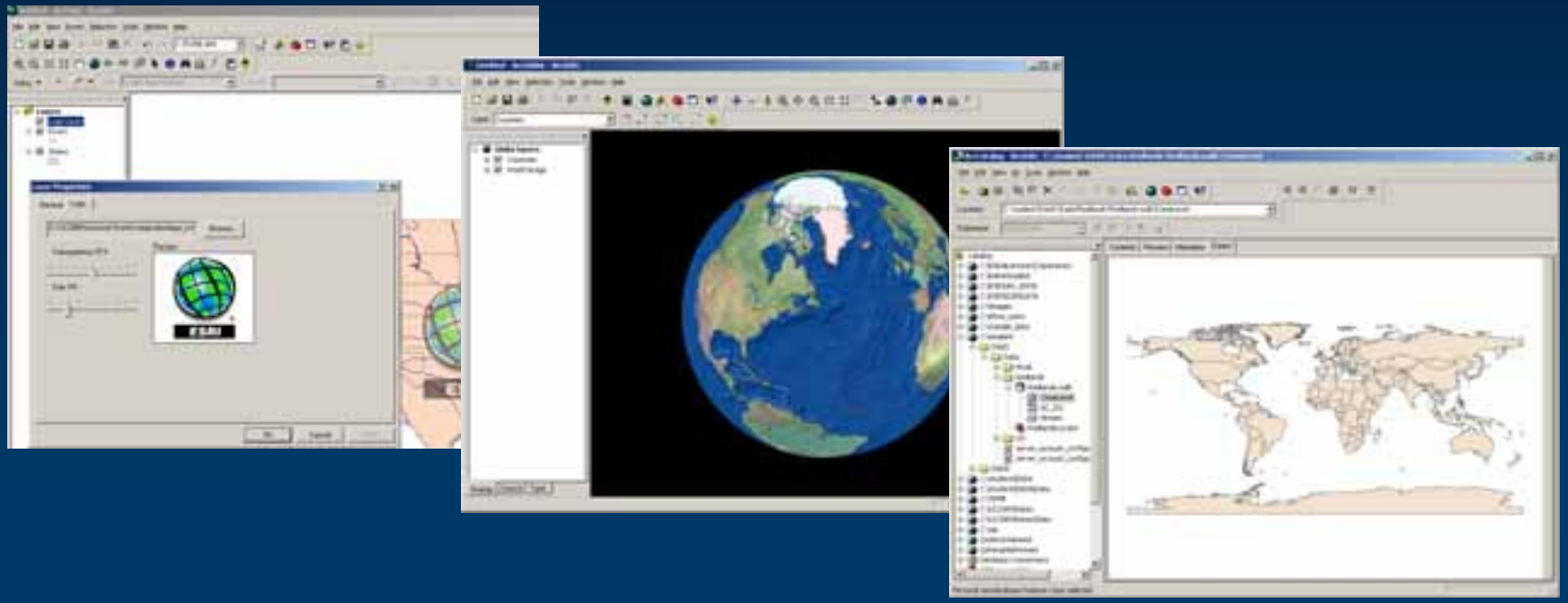
- ArcGIS Desktop
- Installation requirements
- Developer resources
- ArcGIS System architecture
- ArcGIS Desktop application framework
- Implementing custom components
  - Commands, Toolbars, Dockable Windows, Extensions
  - Using IDE integration and BaseClasses

ArcGIS: Many products built with ArcObjects



Extending ArcGIS Desktop applications

- Extend desktop applications by plugging in new custom COM components
- Application framework provides a way to seamlessly integrate your customizations

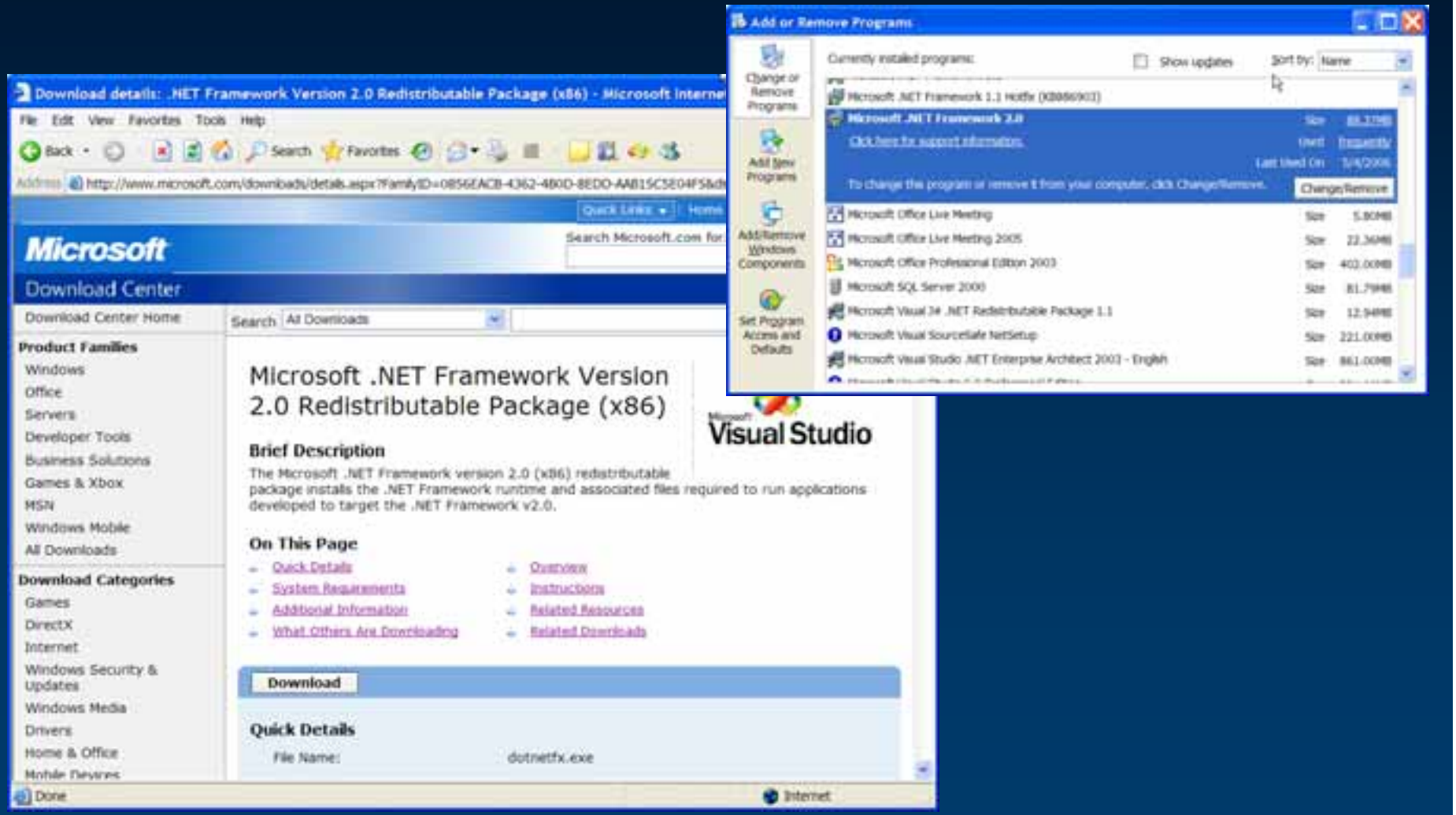


Installation and configuration

- Desktop development requires the following:
  - Installation
    - Product (ArcGIS Desktop)
    - ArcGIS Desktop Developer kit
  - Configuration
    - Developer resources
    - Development environment
  - Understanding product licensing

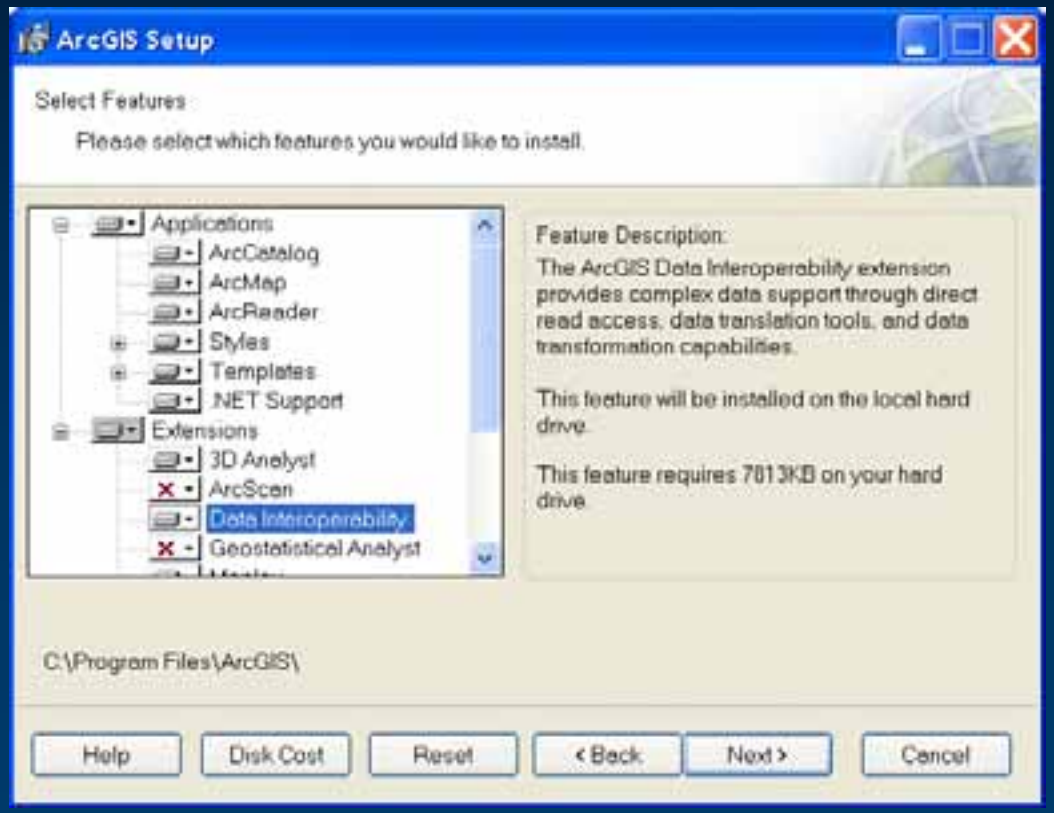
1. Preinstallation – BEFORE installing ArcGIS

- .NET Framework 2.0 required to develop with .NET



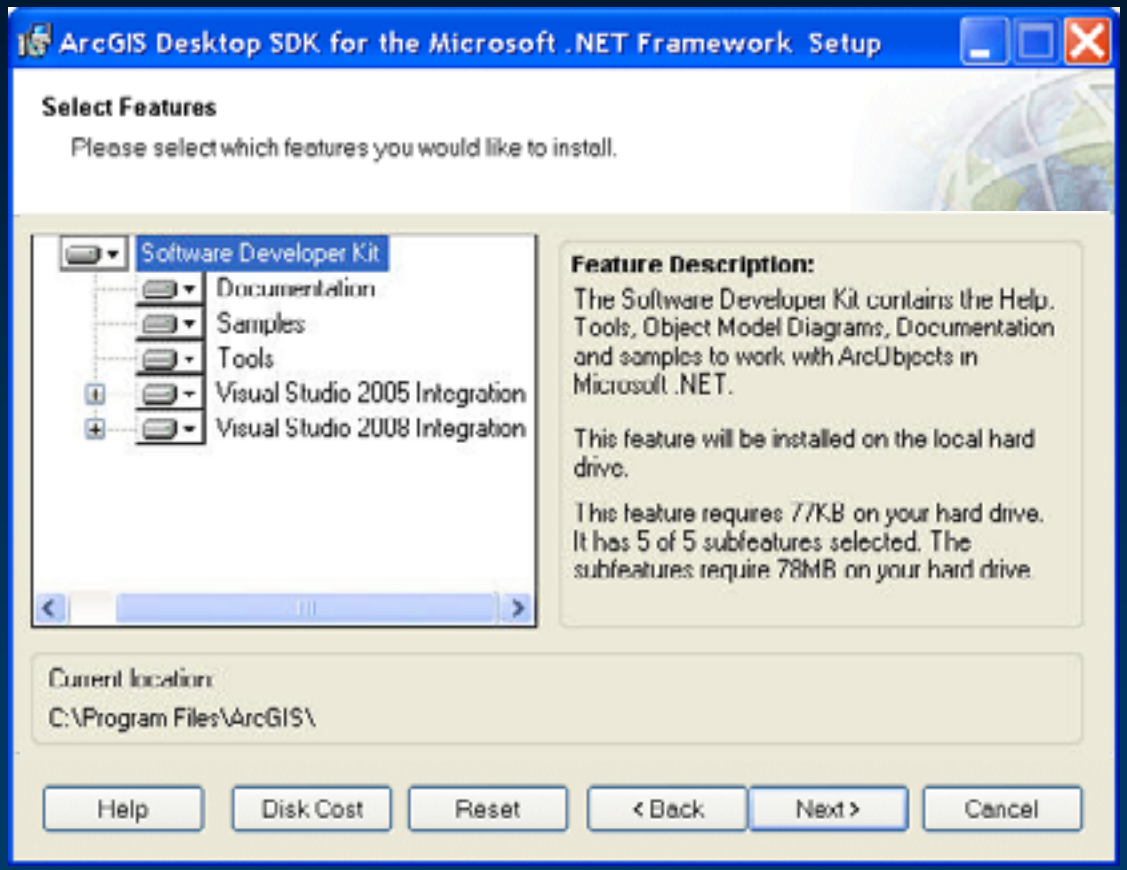
2. ArcGIS Desktop installation

- **ArcGIS applications and extensions**
  - **ArcMap, ArcCatalog, etc.**
  - **May require appropriate license**
- **Core type libraries**
- **Primary interop assemblies**



3. Desktop Developer Kit options

- Different installs for language-specific components
  - COM
  - .NET
  - C++
- Documentation
- Samples
- Tools
- IDE integration components



ArcGIS Desktop product licensing

- Before starting development, understand desktop product licensing
- Content
  - Product has three levels: ArcView, ArcEditor, ArcInfo
  - Many extensions available: ArcGIS 3D Analyst and Spatial Analyst
- Develop with the
  - Version of the software you plan to deploy
  - Level of licensing that you plan to deploy
- Developer tools
  - Desktop Administrator: Check which license is available

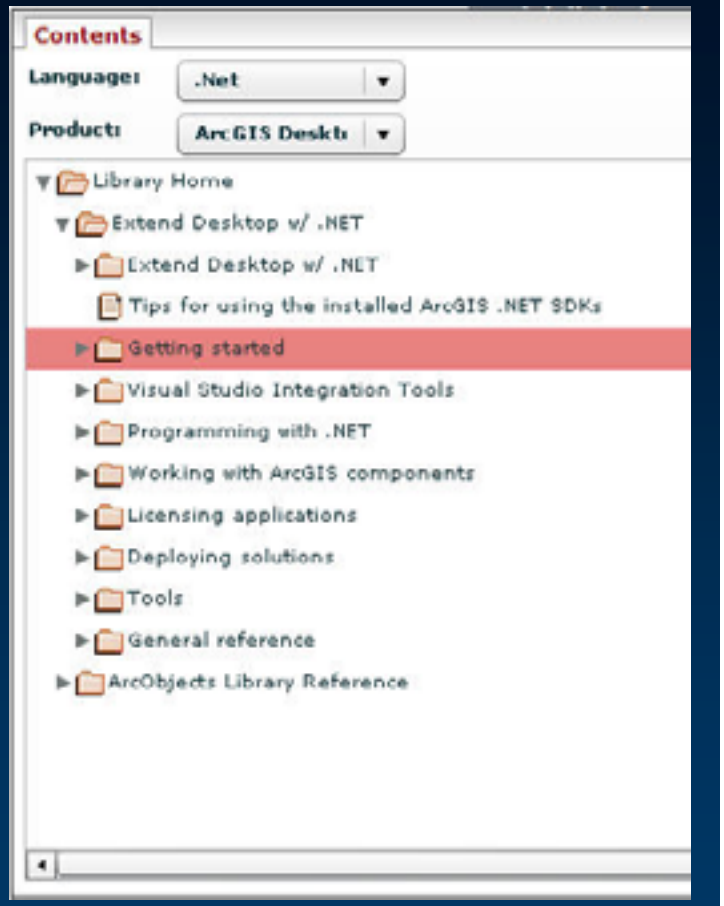
ESRI Developer Network (EDN)

Access to the ArcGIS Development Platform

- **The Developer Product**
  - **ArcGIS Server (all Levels and Editions)**
  - **ArcGIS Image Server**
  - **ArcWeb Services**
  - **ArcIMS**
  - **ArcGIS Engine**
  - **ArcGIS Desktop***
- **Software Developer Kits**
  - **.NET and Java**
- **Website for Developers**
  - **http://edn.esri.com**

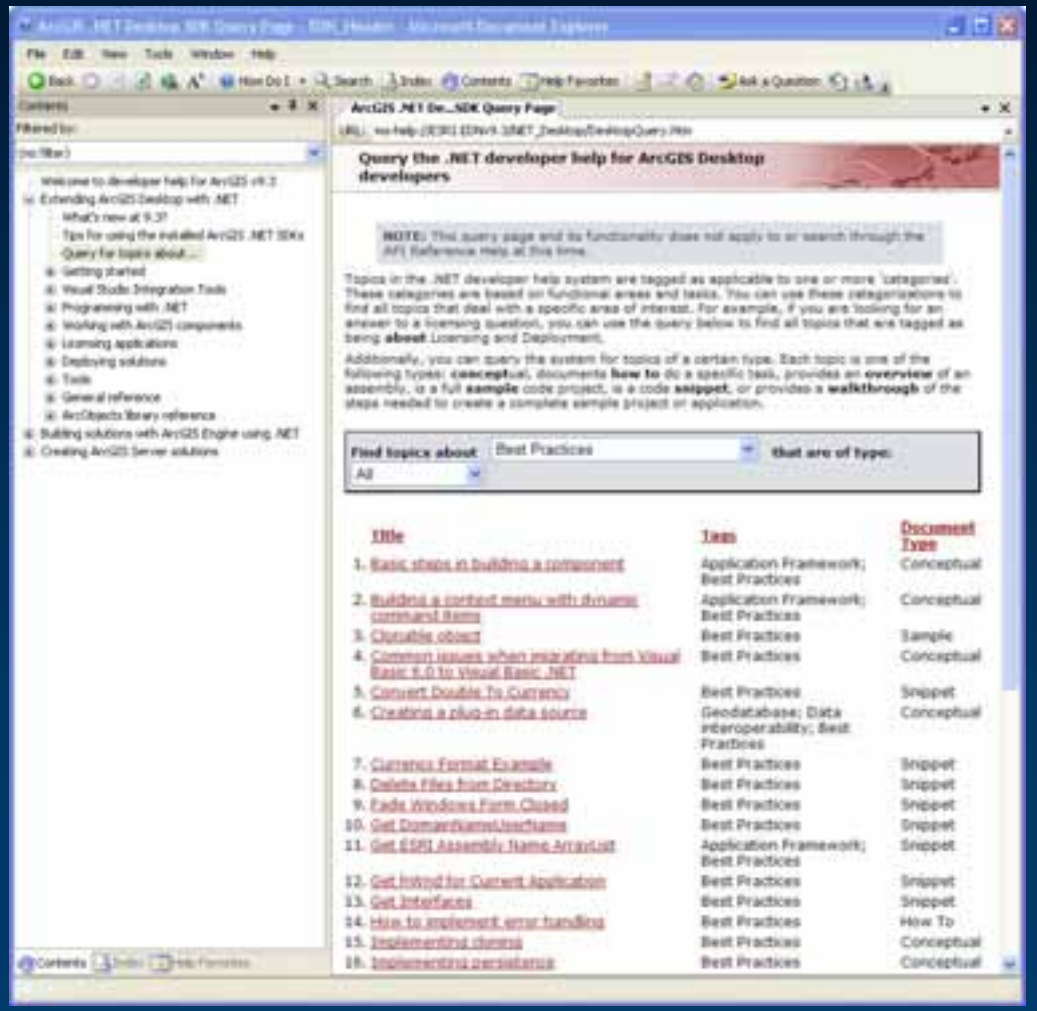
The developer resources

- ArcGIS developer help
  - Language specific help
  - Object model diagrams
  - Books and technical documents
  - Samples
- EDN Website
  - Entire developer help plus
    - Latest developer information
    - Discussion forums
    - Updates, add-ins, and samples



Tip: Querying the developer Help

- Query for topics about…….

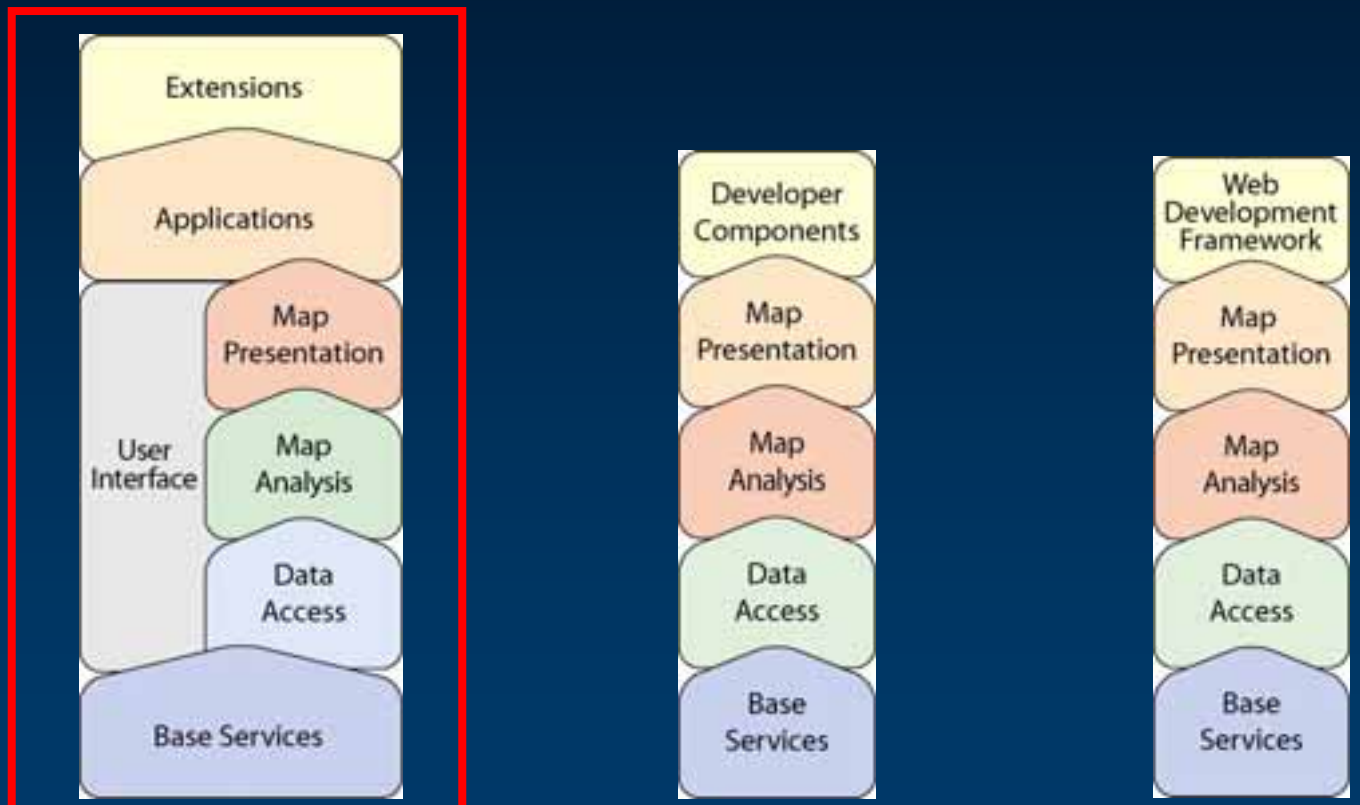


Using ArcObjects in .NET

- ArcGIS architecture
- ArcObjects and the COM API
- ArcObjects and the .NET API
- The Application framework

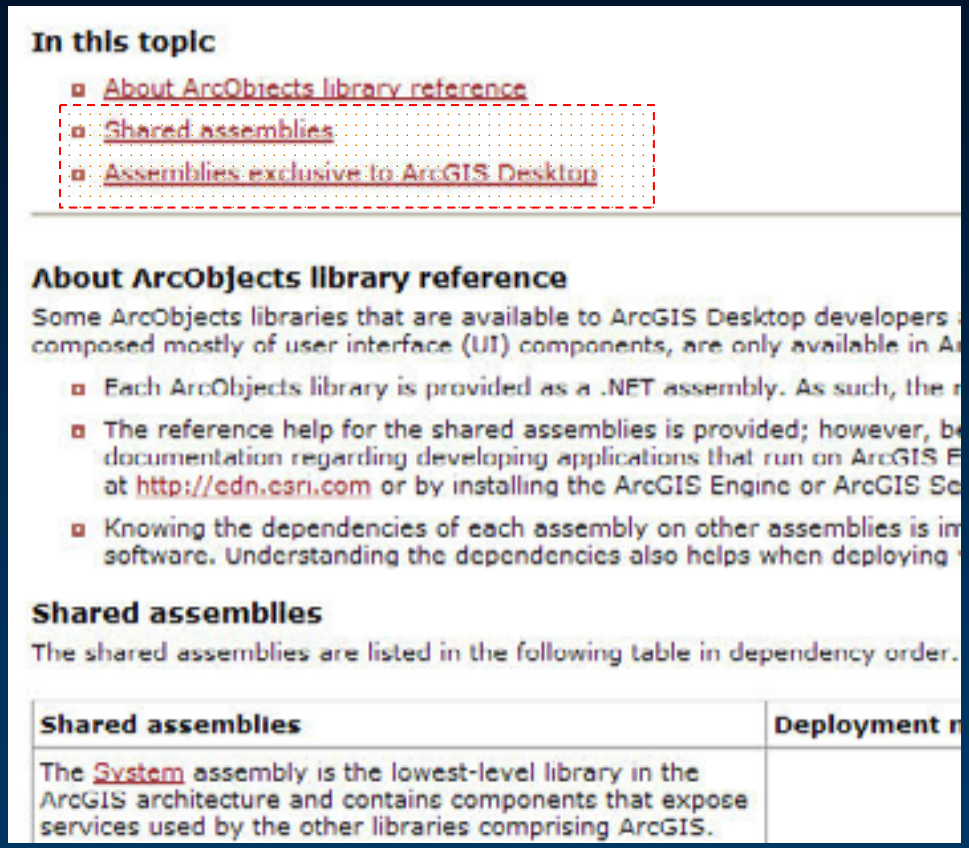
ArcGIS architecture

- All products share ArcObjects
- Desktop extends the core ArcObjects functionality



ArcGIS libraries

- Products share libraries
- *Some libraries are not shared*
  - E.g., esriArcMapUI, Framework
- Identify libraries that belong to different products
- Understand library dependencies
- Developer tools
  - ArcGIS Developer Help
  - Library Locator



In this topic

- About ArcObjects library reference
- Shared assemblies
- Assemblies exclusive to ArcGIS Desktop

About ArcObjects library reference

Some ArcObjects libraries that are available to ArcGIS Desktop developers
composed mostly of user interface (UI) components, are only available in A

- Each ArcObjects library is provided as a .NET assembly. As such, the
- The reference help for the shared assemblies is provided; however, b
documentation regarding developing applications that run on ArcGIS F
at http://edn.esri.com or by installing the ArcGIS Engine or ArcGIS Sc
- Knowing the dependencies of each assembly on other assemblies is in
software. Understanding the dependencies also helps when deploying

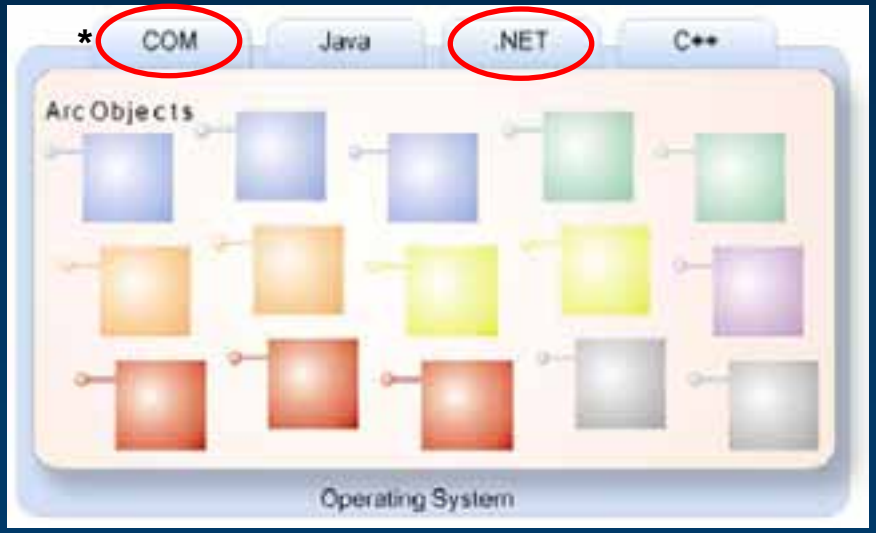
Shared assemblies

The shared assemblies are listed in the following table in dependency order.

Shared assemblies	Deployment r
The System assembly is the lowest-level library in the ArcGIS architecture and contains components that expose services used by the other libraries comprising ArcGIS.	

ArcObjects

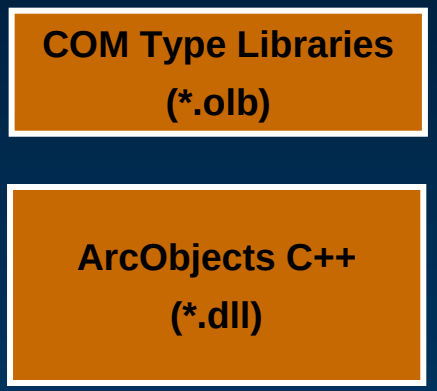
- All ArcGIS products are composed of ArcObjects components
- ArcObjects components are C++ objects
- Accessible through different APIs
- Native interface is COM



*API's supported with ArcGIS Desktop development

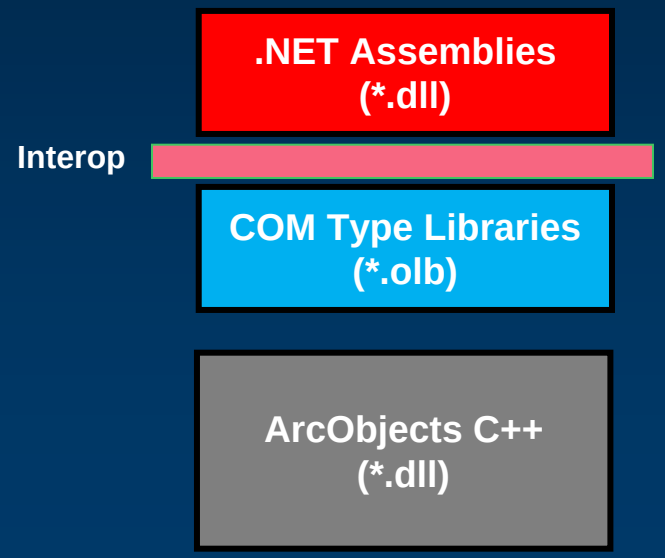
ArcObjects and the COM API

- All ArcObjects COM types are defined in type libraries
- Type libraries are binary files that describe:
  - Classes: CLSIDs and ProgIDs
  - Interfaces: IIDs
  - Constants and enumerations
- Implementation is in DLL and EXE files
- Developer tools
  - ArcGIS Developer Help
  - Regedit, OleView, and EOBrowser



ArcObjects and the .NET API

- All ArcObjects types are defined in .NET assemblies
- Exposes all classes, interfaces, and constants as managed .NET types
- Primary interop assemblies were generated by importing the COM type libraries
- Developer tools
  - ArcGIS Developer Help (.NET)
  - ILDASM.exe



ArcGIS .NET assemblies and namespaces

- All .NET projects must reference the ESRI assemblies
  - Eg., ESRI.ArcGIS.Carto.dll
- Import namespace to access ArcObjects .NET types
  - Eg., ESRI.ArcGIS.Carto
- Assemblies have dependencies

```
'Visual Basic .NET
Imports ESRI.ArcGIS.CatalogUI
Imports ESRI.ArcGIS.ArcMapUI
Imports ESRI.ArcGIS.Framework
. . .
'Visual C#
using ESRI.ArcGIS.CatalogUI
using ESRI.ArcGIS.ArcMapUI
using ESRI.ArcGIS.Framework
```

The application framework

- All applications are built with ArcObjects and have a similar framework
- Customize the user interface at runtime
  - Add/Remove/Reorganize commands, tools, menus, and toolbars
- Customize the applications with VBA
  - Macros, buttons, tools, toolbars, and menus
- Extend the applications by writing a custom COM component and registering it in a component category

ArcGIS Desktop Components

Visual Components (extending the user interface)

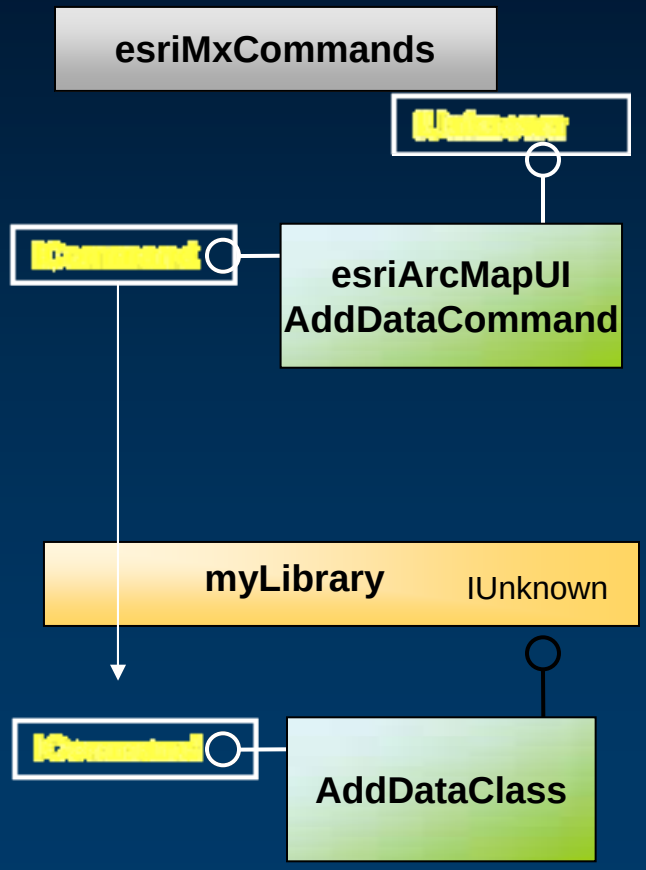
1) Commands and Tools
2) Menus and Toolbars
3) Embedded Windows
   - Dockable Windows, Contents Views, etc.

Non-visual Components

1) Application Extension
2) Undo/Redo Operations
3) Command keyboard shortcuts

Developing custom COM components in .NET

- Create a COM object and plug it into an application
- Steps:
  1. Create a .NET project
  2. Create a COM class
  3. Reference the ArcGIS libraries
  4. Implement an interface
  5. Compile
  6. Register in a component category

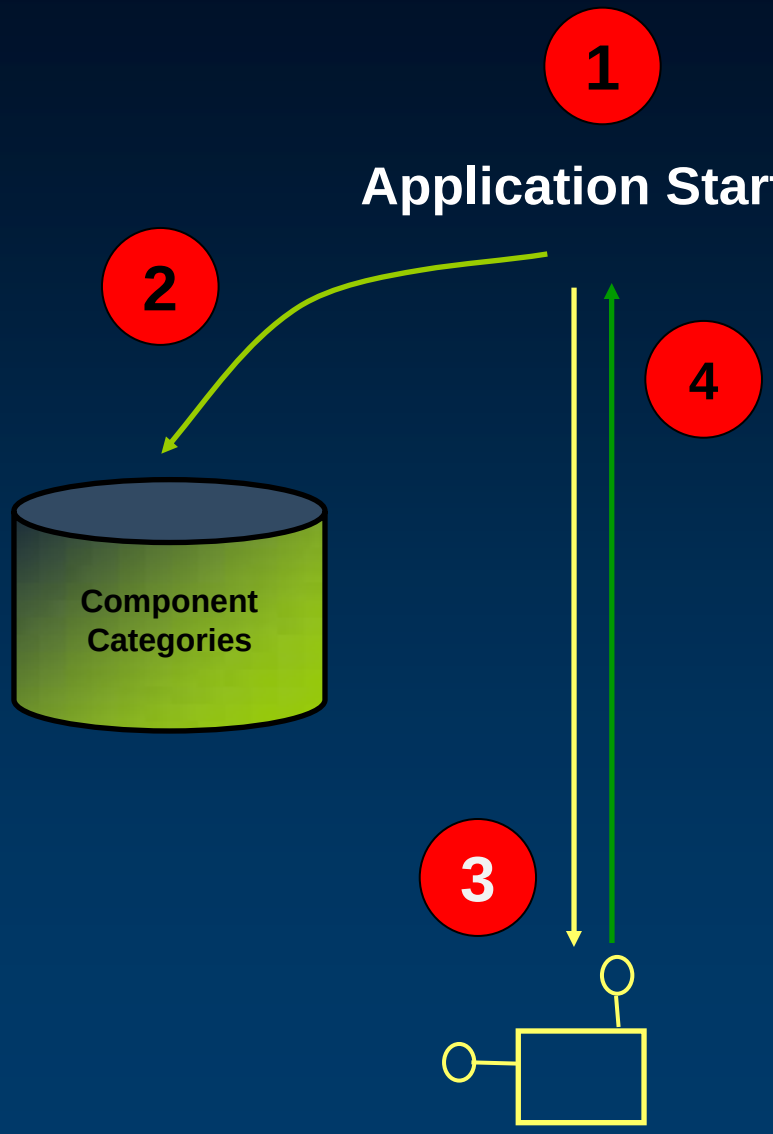


Finding the correct interface

- Identify which interface to implement
- Easiest: model components after existing ESRI objects
- Examples:
  - esriArcMapUI.AddDataCommand - ICommand
  - esriArcMapUI.TOCCatalogView - IContentsViews
- Developer tools
  - Developer Help: OMDs and samples
  - Extending ArcObjects
  - EOBrowser and Oleview

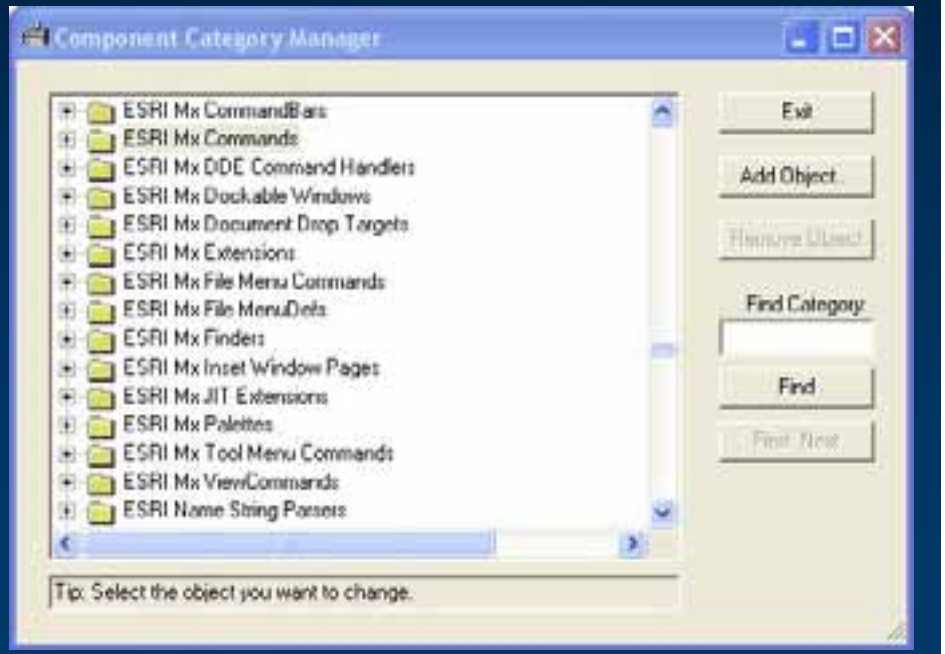
Application startup cycle

- Components loaded at runtime
- Steps:
  1. Application starts
  2. Reads the component categories from the registry
  3. Creates all ESRI and custom components
  4. Loads the objects



Component Categories

- Registry folders
  - Out of the box components
  - Read by the Customize Dialog
  - Register custom components
- ArcGIS Desktop categories
  - ESRI Mx Commands
  - ESRI Mx CommandBars
  - ESRI Mx Dockable Windows
  - ESRI Mx Extensions
  - Many others……



Finding ArcGIS Desktop components

- SDK Documentation
- General > Names and ID's
  - ArcMap, ArcCatalog,ArcGlobe, ArcScene Commands
  - Extensions

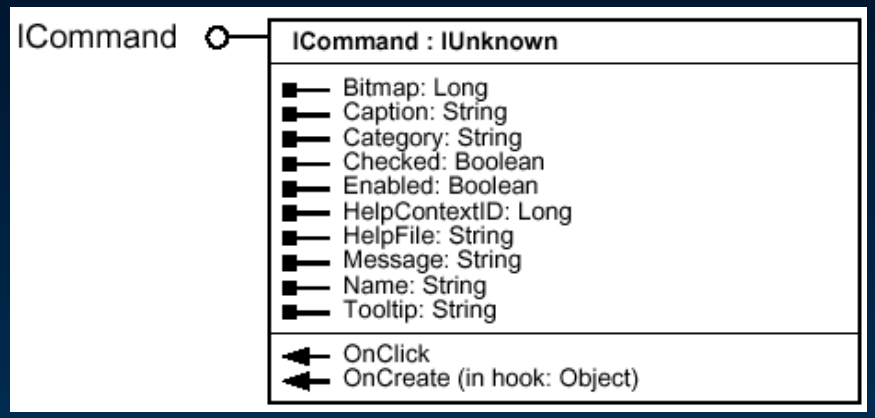
Type	Caption	Name	Command Category	GUID (CLSID / ProgID)
Toolbar	Main Menu	Main Menu	none	{1E739F59-E45F-11D1-9496-080009EEBECB} esriArcMapUI.MxMenuBar
Menu	File	File_Menu	none	{56599DD3-E464-11D1-9496-080009EEBECB} esriArcMapUI.MxFileMenu
Command	New	File_New	File	{119591D8-0255-11D2-8D20-080009EE4E51} esriArcMapUI.MxFileMenuItem
Command	Open	File_Open	File	{119591D8-0255-11D2-8D20-080009EE4E51} esriArcMapUI.MxFileMenuItem
Command	Save	File_Save	File	{119591D8-0255-11D2-8D20-080009EE4E51} esriArcMapUI.MxFileMenuItem
Command	Save As	File_SaveAs	File	{119591D8-0255-11D2-8D20-080009EE4E51} esriArcMapUI.MxFileMenuItem
Command	Save A Copy	File_SaveCopyAs	File	{119591D8-0255-11D2-8D20-080009EE4E51} esriArcMapUI.MxFileMenuItem
Command	Add Data	File_AddData	File	{E1F29C6B-4E6B-11D2-AE2C-080009EC732A} esriArcMapUI.AddDataCommand
Menu	Add Data From GIS Portal	AddInternetData_Menu	none	{5B43EFCE-8C6F-49FF-8B77-0C769AB4A5F0} esriArcMapUI.AddInternetDataMenu

Commands: Framework/Background

- Application loads your class and creates the command
- Default implementation is provided
  - Button window frame
  - Handling of events
  - Ability to customize
  - Persisting to document
- You provide basic implementation
  - Name of the command
  - Click event
  - Tooltip
  - Icon
  - etc.

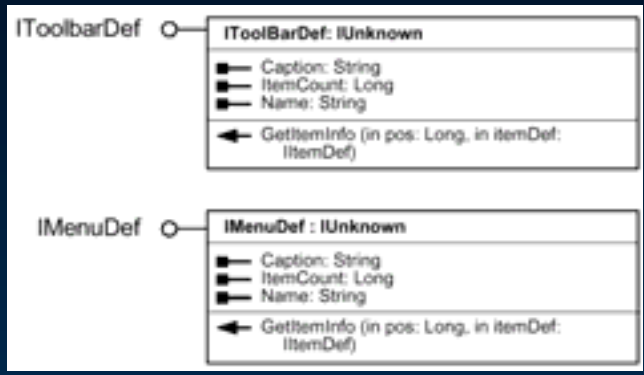
Commands: Implementation details

- Model after the Button class
- Interface: ICommand
- Base Class: BaseCommand
- Type Library: esriSystemUI
- Component Category
  - ESRI Mx Commands: ArcMap
  - ESRI Gx Commands: ArcCatalog
  - ESRI Sx Commands: ArcScene
  - ESRI GMx Commands: ArcGlobe
- Other related interfaces and objects
  - ICommandItem
  - ITool, IToolControl



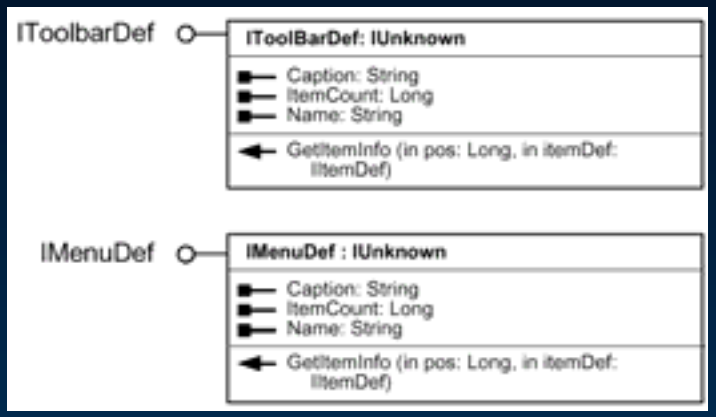
Toolbars: Implementation details

- Model after an ESRI toolbar
- Interfaces: IToolBarDef
- Base Class: BaseToolbar
- Type Library: esriSystemUI
- Component Category
  - ESRI Mx CommandBars: ArcMap
  - ESRI GX CommandBars: ArcCatalog
  - ESRI Sx CommandBars: ArcScene
  - ESRI GMx CommandBars: ArcGlobe
- Other related interfaces and objects
  - Buttons, Tools
  - Extensions



Menus: Implementation details

- Model after an ESRI menu
- Interfaces: IMenuDef
- Base Class: BaseMenu
- Type Library: esriSystemUI
- Component Category
  - ESRI Mx CommandBars: ArcMap
  - ESRI GX CommandBars: ArcCatalog
  - ESRI Sx CommandBars: ArcScene
  - ESRI GMx CommandBars: ArcGlobe
- Other related interfaces and objects
  - IRootlevelMenu, IShortcutMenu, IMultiItem

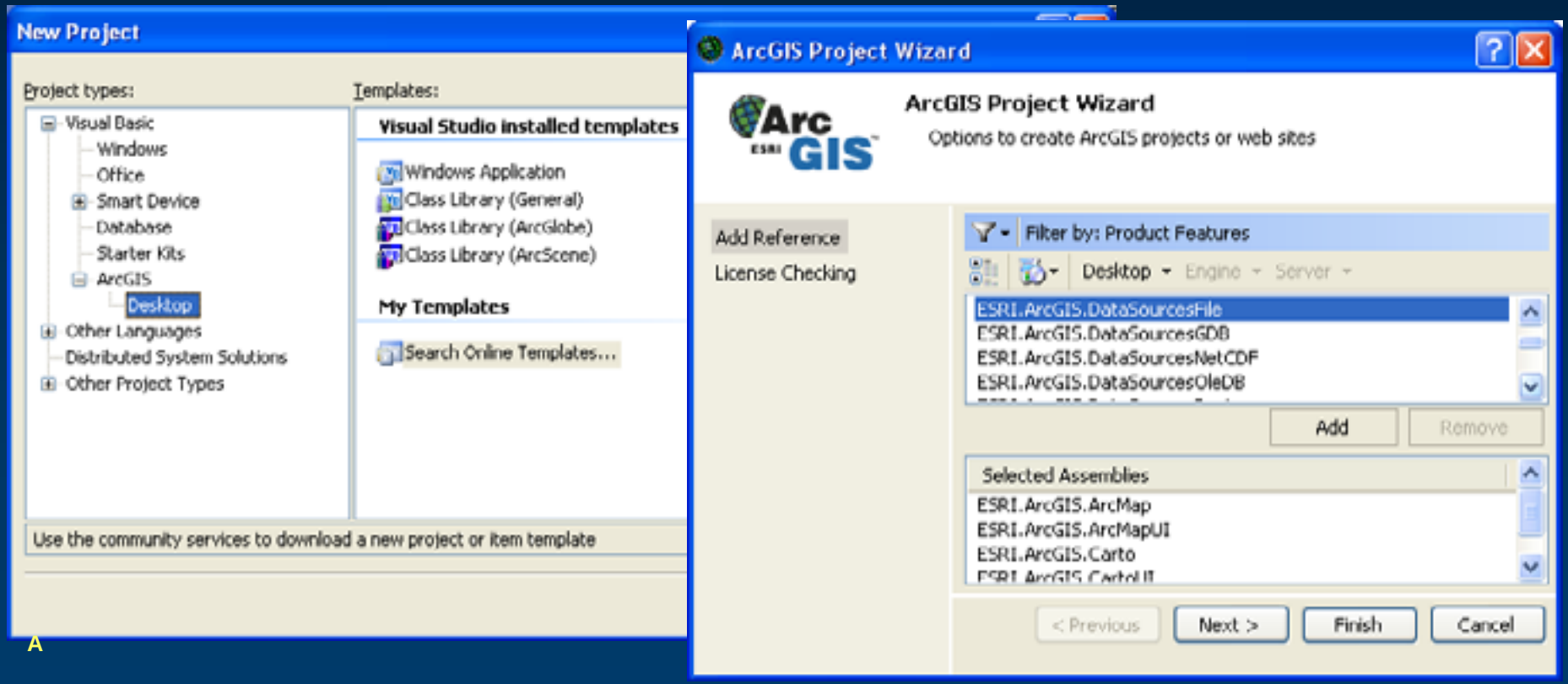


Development tools and shortcuts

- New Project templates
- Base classes
  - BaseCommand,BaseTool,BaseToolbar,BaseMenu
- Add Class wizard
- Code Snippets
  - Snippet Finder Utility
- Quickly adding Imports/Using statements

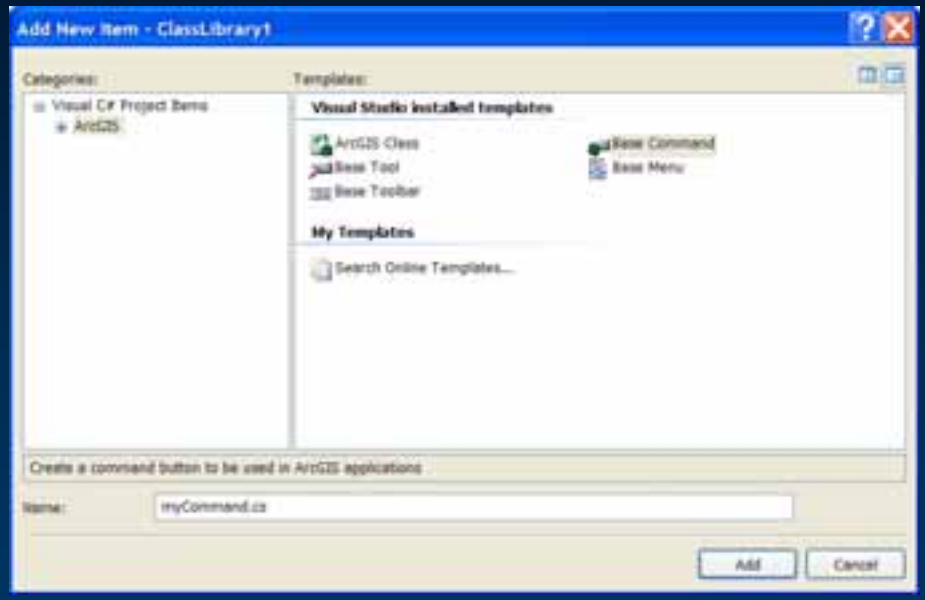
.Net new project templates

- Available when new VS.Net project is started
- After a template is chosen, ArcGIS Project Wizard opens allowing users to select ArcGIS References



Using .NET Base Classes

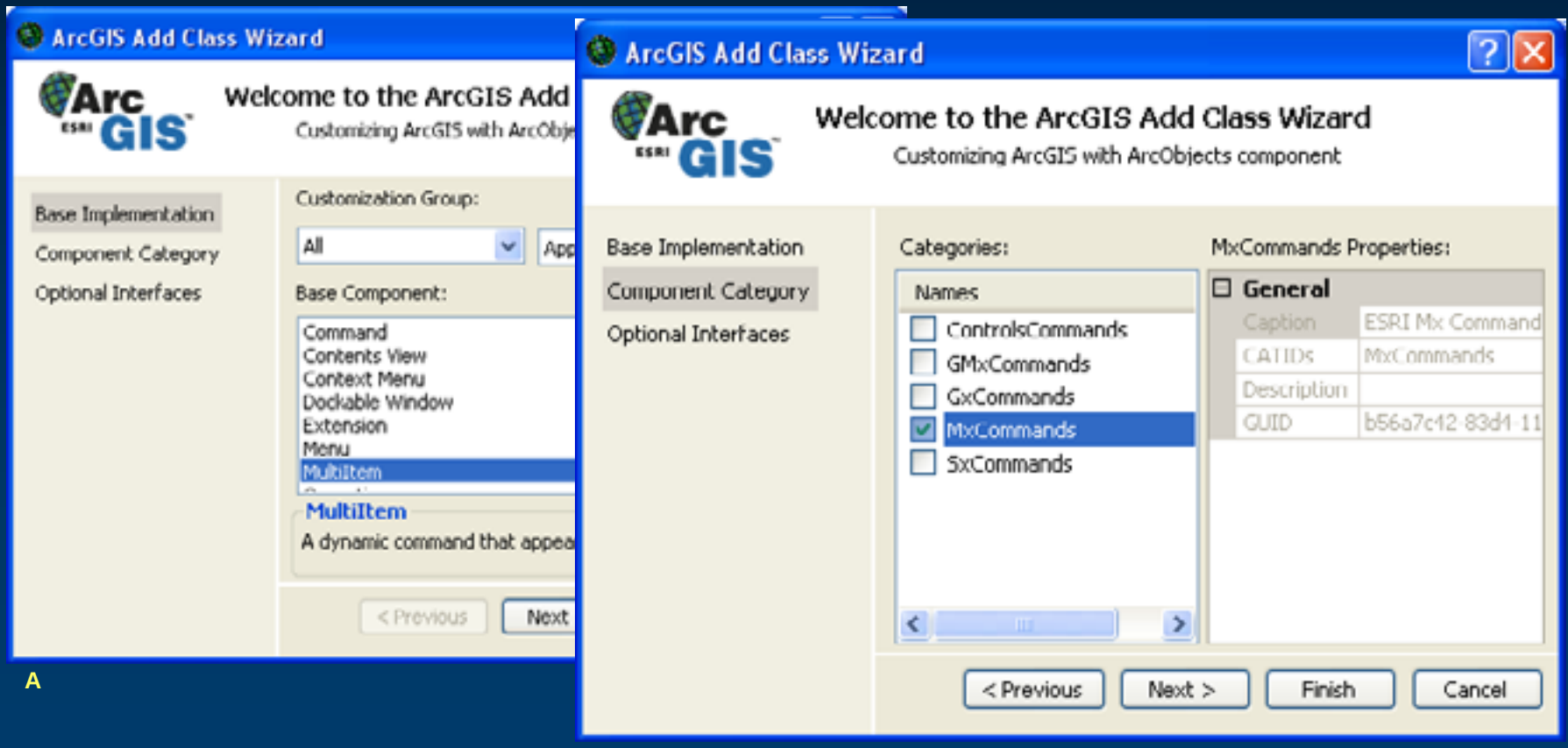
- Available .NET base classes
  - BaseCommand
  - BaseTool
  - BaseToolbar
  - BaseMenu
  - BaseCustomLayer
  - BaseGlobeCustomLayer



- Easy to use
  - ESRI.ArcGIS.ADF.BaseClasses assembly
  - Use ArcGIS Visual Studio Integration Framework

Add Class Wizard

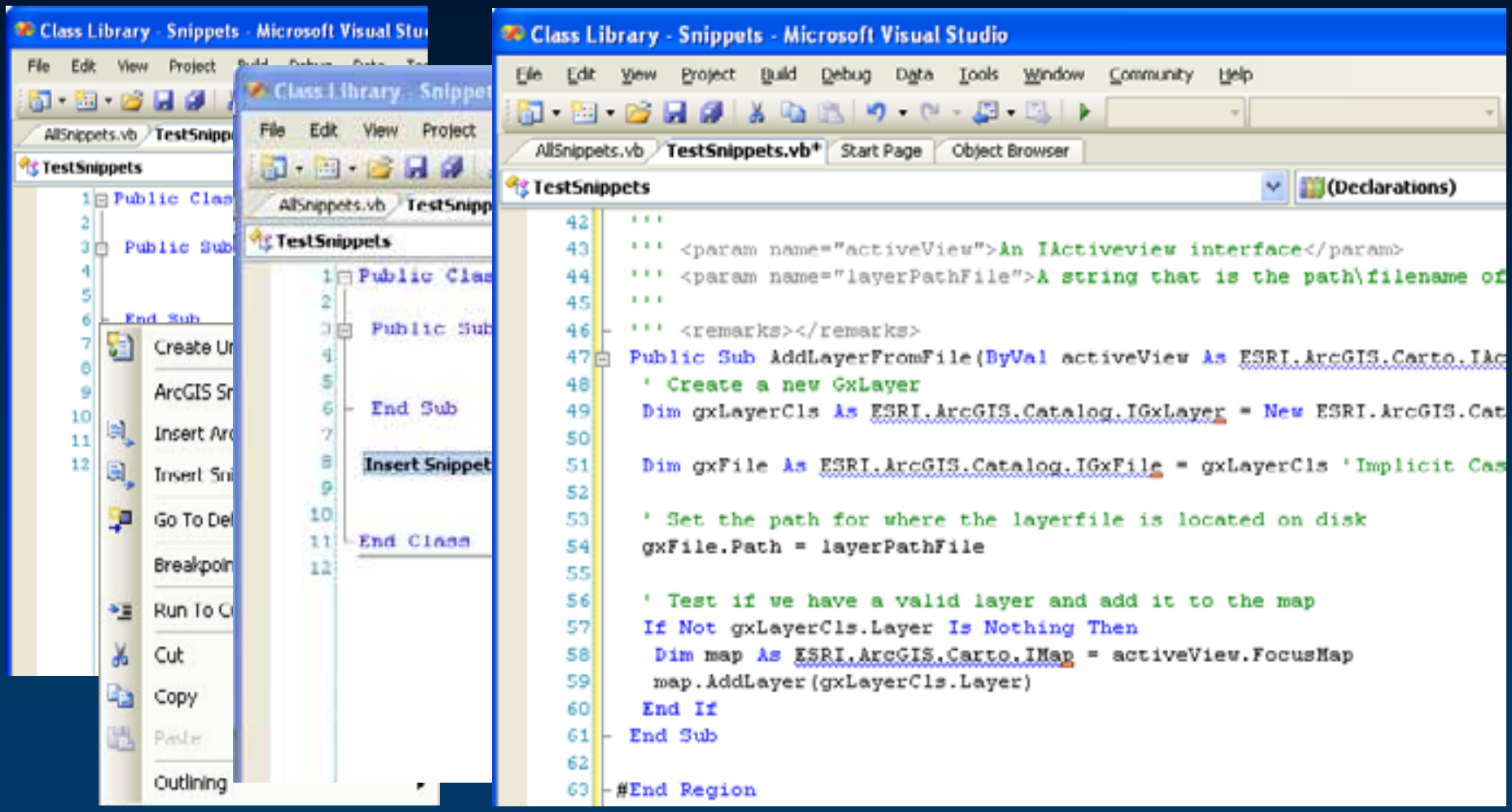
- Allows users to add classes to .Net that automatically add a specific interface/base class
- Component Category registration code added to class



A

.Net code snippets

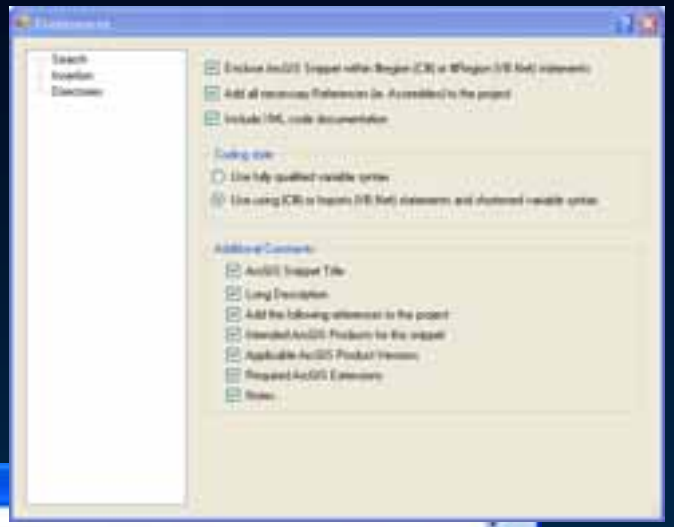
- **Insert Snippet – Reuseable ArcObjects code**

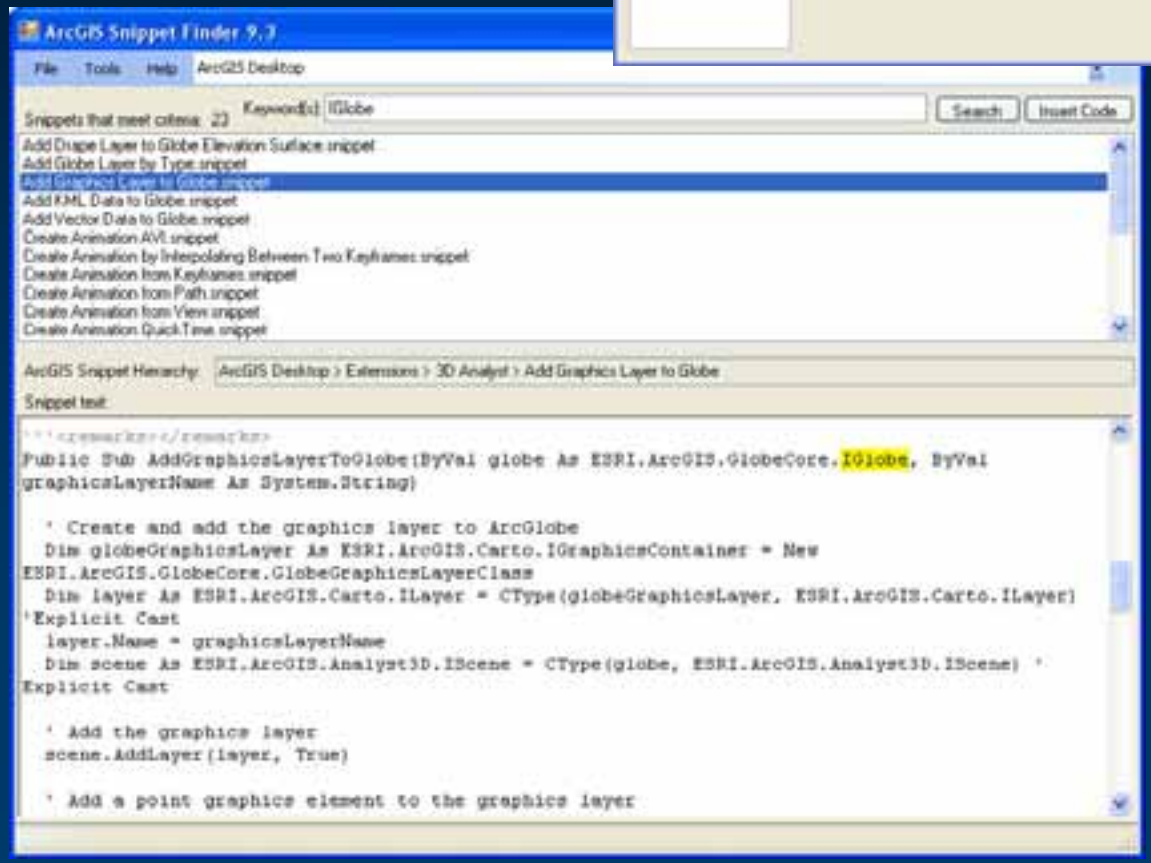


A

ArcGIS Snippet Finder

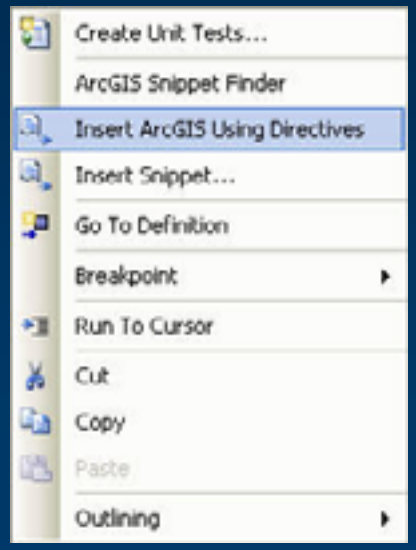
- Integrated within Visual Studio
- Keyword search of available snippets
- Ability to set preferences for how search and code insertion works





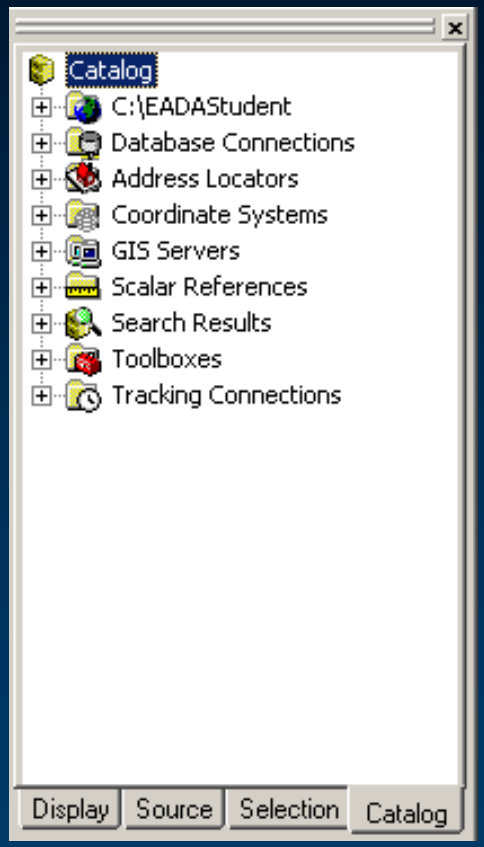
Add import/using statements

- ArcGIS Reference dialog box allows users to quickly add appropriate ESRI references
- Use Namespaces tab to make specific references
  - Allows right-click in VS and add Imports/using directives that are referenced in Namepaces tab without writing code



Custom Windows: Plugging a form into the application

- Three main types
  - Property pages
  - Tabbed views
  - Dockable windows

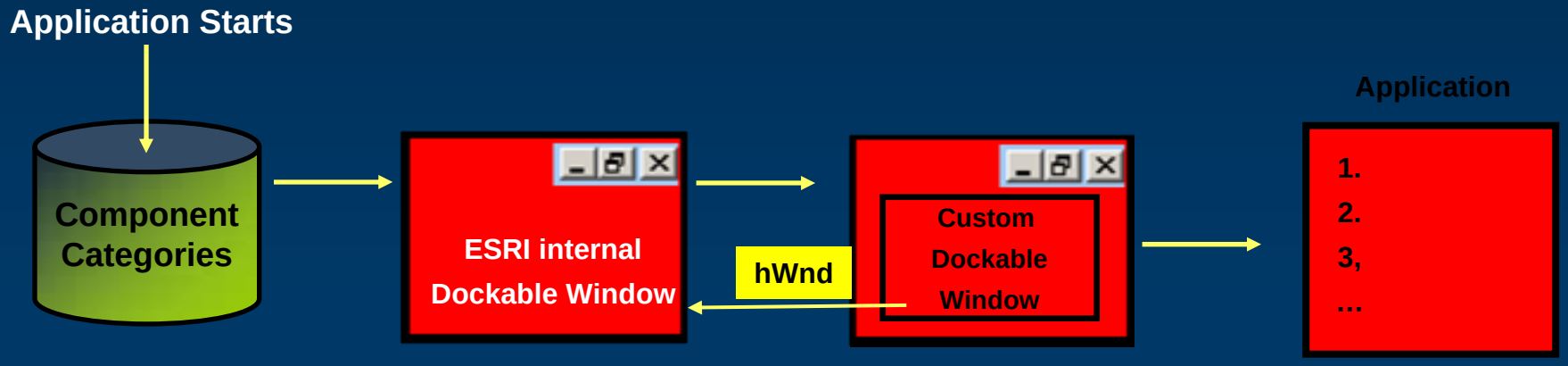


ArcGIS windows application framework

- Provides a mechanism to plug in forms or parts of forms
- Manages
  - Parent frame of the window
  - Placement of windows
  - Resizing
  - User events
  - Persisting properties of windows
- Provides internal classes with default implementation
  - Dockable window, contents view, property sheets
- User provides behavior of internal child window

Loading custom windows

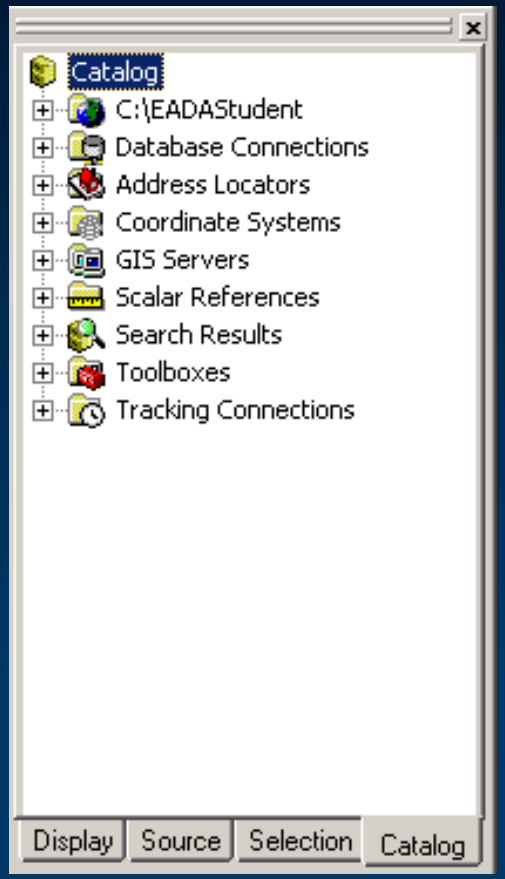
- Automatically loaded by the application or parent window
- Basic loading sequence is the same
  1. Access the appropriate component category
  2. Create an internal window object
  3. Create your custom child window
  4. Windows are added to a list maintained by the application



E.g., ESRI Mx Dockable Windows

Parent and child windows

- Parent windows are frames that contain child windows
- Parent windows control
  - Position, redrawing, and messages
- Child window contains controls
  - Buttons, list boxes, ActiveX controls
- Child window has ArcObjects code
- Need to manage lifetime explicitly
  - Create and Destroy
  - Activate and Deactivate
  - Show and Hide



Dockable windows

- Floating or docked window that is associated with the application
- Show with a command
- Available for all applications
- Examples
  - ESRI TOC window
  - Show any type of data
    - Symbols, charts, and metadata
  - May contain ESRI controls

IDockableWindowDef: Main properties

- Properties that identify the window, including name and caption
- Can store any related data with the window
  - UserData: Way to hook back into child window objects, e.g., MapControl, list box, texbox

```
Public ReadOnly Property Name() As String Implements IDockableWindowDef.Name
        Get
            Return "Selection Count"
        End Get
End Property
Public ReadOnly Property Caption() As String Implements IDockableWindowDef.Caption
        Get
            Return "Selected Features Count"
        End Get
End Property
 Public ReadOnly Property UserData() As Object Implements IDockableWindowDef.UserData
    Get
            Return m_dockWinForm.MapControl
    End Get
End Property
```

IDockableWindowDef: Properties

- Window management properties
  - OnCreate: Create window, Set Hook, Sink events
  - OnDestroy: Destroy window
  - ChildHWND

```
Public Sub OnCreate(ByVal hook As Object) _
    Implements IDockableWindowDef.OnCreate
    m_dockWinForm = New DockWinFormVBNet
    ' Establish a hook to the application
    m_app = hook
    SetupEvents() 'Call routine to sink events
    OnSelectionChanged()
End Sub
Public Sub OnDestroy() Implements IDockableWindowDef.OnDestroy
    m_app = Nothing
End Sub
Public ReadOnly Property ChildHWND() As Integer _
    Implements IDockableWindowDef.ChildHWND
    Get
      ' Dockable window is a listbox.  Pass back the hWnd of this listbox
      ChildHWND = m_dockWinForm.ListSelCount.Handle.ToInt32
    End Get
End Property
```

DockableWindows: Associating with commands

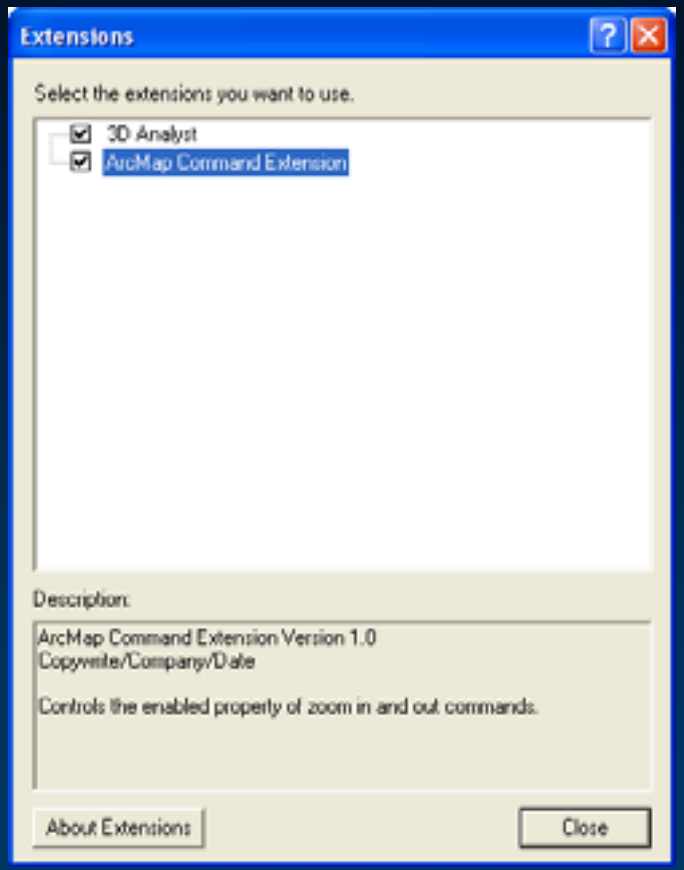
- Create a custom command to manage the window
  - OnCreate: Find the window, hold on to window reference
  - OnClick: Show or hide the window
- Need to know the CLSID of the dockable window
- Windows are created before commands during startup

```
'Implementation of ICommand
Public Overrides Sub OnCreate(ByVal hook As Object)
      m_pApp = hook
      m_pDockWinMgr = hook 'QI for IDockableWindowManager
      Dim u As New UID
      u.Value = "DockableWindow.DockWin"   'Reference to dockable window
      m_pDockWin = m_pDockWinMgr.GetDockableWindow(u)
End Sub

Public Overrides Sub OnClick()
    ' Toggle the visibility of the dockable window
    m_pDockWin.Show(Not m_pDockWin.IsVisible)
End Sub
```

ArcGIS application framework for extensions

- Mechanism to plug objects into the application
- Accessible to all objects
- Generally, visible through Extension Manager
- Supported by all applications
- Examples:
  - Editor extension, 3D Analyst
  - Any custom extension desired



Application extensions: Framework/Background

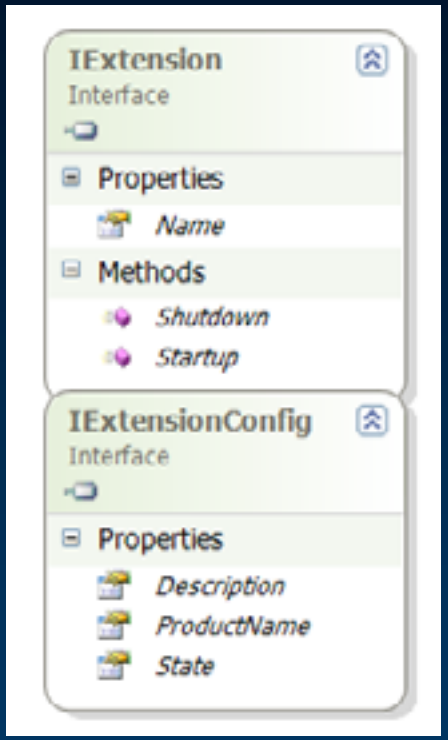
- Two types of extensions
- Invisible
  - Automatically enabled by application at runtime (e.g., Editor)
  - Transparent to user that extension has been loaded
- Visible
  - Enabled through extension manager window
  - Generally require a license or special criteria to enable
  - Some check out an ESRI license (e.g., 3D Analyst)

Application extensions: Framework/Background

- Application loads extensions automatically at startup
- Loaded early in the startup cycle
- Developer responsible for implementing behavior of what happens at startup and shutdown
  - Reference the application
  - Sink document events
  - Persist (load and save) data into the document
  - Check out a license for other components
  - Configure data that other components will access

Application extensions: Implementation details

- Model after any ESRI Extension
- Interface(s):
  - IExtension, IExtensionConfig
- Type Library: esriSystem
- Component Category
  - ESRI Mx Extensions
  - ESRI GX Extensions
  - ESRI Sx Extensions
  - ESRI GMx Extensions



IExtension: Invisible extensions

- Required to implement three members
  - Name: Internal name
  - Startup: Passed in the application or other data
  - Shutdown: Clean up all references

```
Private m_app as IApplication

Public ReadOnly Property Name() As String Implements IExtension.Name
        Get
            Return "My Custom Extension"
        End Get
End Property

Public Sub Shutdown() Implements IExtension.Shutdown
        m_app = Nothing
End Sub

Public Sub Startup(ByRef initializationData As Object) Implements IExtension.Startup
        If (TypeOf initializationData Is IMxApplication) Then
            m_app = initializationData
        End If
End Sub
```

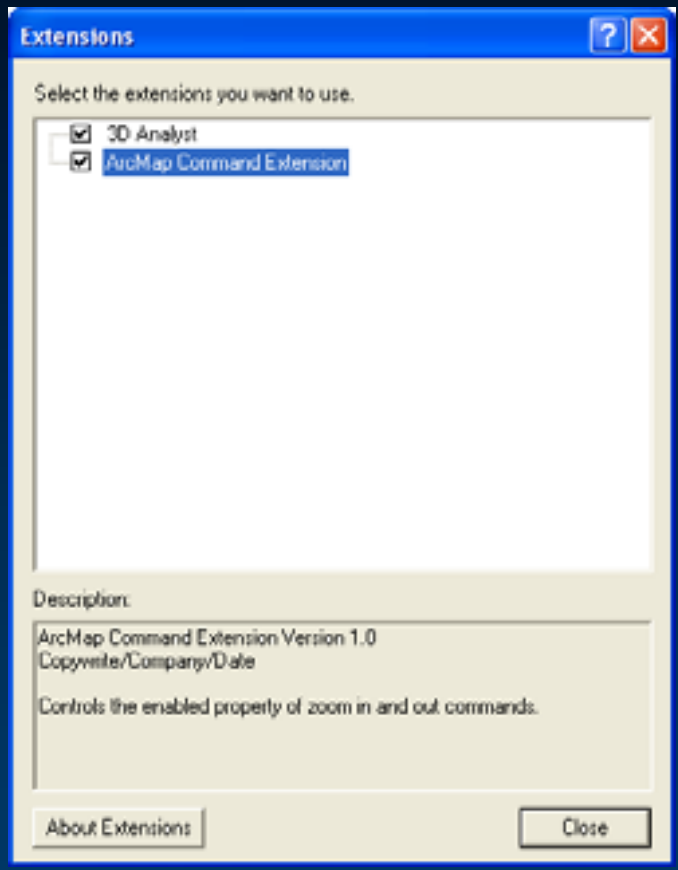
Custom extensions: Document events

- Use Startup to sink document events
  - NewDocument, OpenDocument, CloseDocument
  - ActiveViewChanged, MapsChanged, OnContextMenu

```
Private documentEvents As MxDocument
Private dNewDocument As IDocumentEvents_NewDocumentEventHandler
Public Sub Startup(ByRef initializationData As Object) Implements IExtension.Startup
    If (TypeOf initializationData Is IMxApplication) Then
        m_app = initializationData
        documentEvents = CType(m_app.Document, MxDocument)
        SinkEvents()
    End If
End Sub
Private Sub SinkEvents()
    dNewDocument = New IDocumentEvents_NewDocumentEventHandler(AddressOf OnNewDoc)
    AddHandler CType(documentEvents, IDocumentEvents_Event).NewDocument,
dNewDocument
End Sub
Public Sub OnNewDoc()
    ' Do some sort of check
    If (Not ValidUser) Then DisableSomeCommands
End Sub
```

IExtensionConfig: Visible extensions

- Adds extension to the extension manager window
- Name appears in the list
- Looks like all other ESRI extensions
- Description appears at the bottom
  - Extension name and version
  - Copyright and company name
  - Extension purpose



IExtensionConfig: Extension state

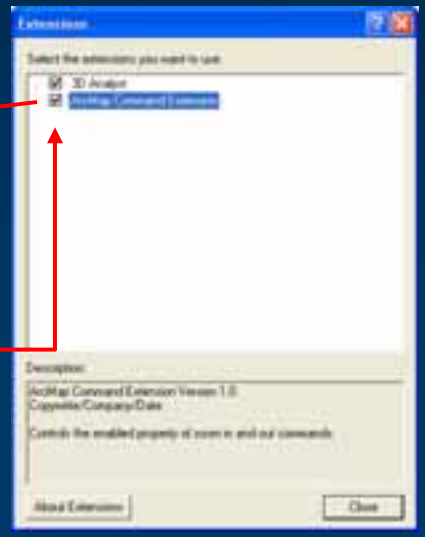
- Three different states are possible
  - esriESEnabled: License available and is checked
  - esriESDisabled: Is not checked
  - esriESUnavailable: License is not available
- Extension state is persisted in the registry
  - HKEY_CURRENT_USER\Software\ESRI\ArcMap\Extensions

Custom extensions: Managing state

- Synchronize checkbox state with the extension
- Store value in a member variable in the extension
- Get and set value at various times
  - User interacts with checkbox to start and shut down extension

```
Private m_iExtensionState As esriExtensionState

Public Property State() As ESRI.ArcGIS.esriSystem.esriExtensionState
Implements ESRI.ArcGIS.esriSystem.IExtensionConfig.State
        Get
            'Read the extension state
            Return m_iExtensionState
        End Get
        Set(ByVal value As ESRI.ArcGIS.esriSystem.esriExtensionState)
            'Write the extension state
            m_iExtensionState = value
        End Set
End Property
```



A

Custom extensions: Controlling commands

- Extension is checked: Enable command
- Extension is not checked: Disable command

```
' Extension class
Public Property State() As ESRI.ArcGIS.esriSystem.esriExtensionState
Implements ESRI.ArcGIS.esriSystem.IExtensionConfig.State
        Get
            'Read the extension state
            Return m_iExtensionState
        End Get
        Set(ByVal value As ESRI.ArcGIS.esriSystem.esriExtensionState)
            'Write the extension state
            m_iExtensionState = value
        End Set
End Property
```

```
' Command class
Public Overrides ReadOnly Property Enabled() As Boolean
        Get
            If (m_extensionConfig.State = esriExtensionState.esriESEnabled) Then
                Return True
            Else
                Return False
            End If
        End Get
End Property
```

A

Custom extensions: Finding extensions

- All objects can find and reference extensions
- You can store common data in your own properties

```
'Command class
Private m_extensionConfig As IExtensionConfig
Public Overrides Sub OnCreate(ByVal hook As Object)
    If Not (hook Is Nothing) Then
       If TypeOf (hook) Is IMxApplication Then
            m_application = CType(hook, IMxApplication)
            Dim uid As New UID
            uid.Value = "Lecture5.Lect5Extension"
            Dim extension As IExtension
            extension = m_application.FindExtensionByCLSID(uid) 'Find extension
            m_extensionConfig = extension 'QueryInterface
        End If
    End If
End Sub
```

To learn more…

- Other recommended sessions and meetings
  - Deploying Desktop Applications
    - 1:00-2:15 (W) Pasadena/Sierra/Ventura
- Demo theaters
  - Using the ArcGIS Carto, Display, and Output APIs
    - 3:00pm (W)
  - Working with Geocoding APIs
    - 4:00pm (W)

In Conclusion…

- Slides and code will be available on EDN
- Please fill out session surveys!
- Still have questions?
  1. Tech talk, Demo Theatres, Meet the Team
  2. “Ask a Developer” link on web page