Migrating VBA/VB6 ArcObjects Applications to .NET

Don Kemlage, Jianxia Song and Jeremy Wright

Schedule

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

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

Introductions

- Who are we?
- Who are you?

Our purpose today

Give practical experience in migrating ArcObjects code from VBA/VB6 to .NET

- Why Migrate
- Key Concepts
- Language Differences
- Visual Studio Migration Features
- Bringing It All Together
- Additional Resources

Why Migrate

It depends on your situation

- If it ain’t broke don’t fix it
- Stay competitive

It will be a challenge

- Costly
- 7,000 – 10,000 lines of code converted per week
- 2 to 3 weeks understanding all migration issues
- Understand the .NET Framework capabilities

Why Migrate

VBA/VB6 disadvantages

- Falling behind the technology curve
- Not a ‘true’ object oriented language
- Have to call Win32 API to access system internals
- IDE issues
  - Debugging problematic for COM objects
  - VB magic
- DLL hell

Why Migrate

The Facts

VBA

- As of July 1, 2007 Microsoft discontinued new VBA distribution licenses
- ESRI VBA SDK's are in maintenance mode
- Note: ArcGIS 9.3 will be using VBA version 6.5

VB6

- Microsoft will end support at the end of 2008
- ESRI VB6 SDK's are being deprecated in 9.3

Why Migrate

.NET Advantages

- True Object Oriented language
  - Supports Inheritance, function overloading, structured exception handling, multithreading, garbage collection
- .NET Framework
  - Common Language Runtime (CLR)
    - Virtual machine, MSIL
  - The .NET Framework class library

Why Migrate

.NET Advantages (continued)

- Visual Studio IDE
  - One IDE for all .NET projects
  - Option to choose from Multiple Programming Languages
  - Other neat features

  - ESRI’s Visual studio integration features
  - Easy Debugging ArcObjects application

Why Migrate

.NET Advantages (continued)

- Easy Deployments
  - No more .dll hell
  - XCOPY .NET Assemblies to target machine
  - Recompile not necessary to target different Operating Systems
- Improved maintenance and scalability

Key Concepts

- COM API
  - Expose type information using type libraries
- .NET API
  - Use metadata as a source of type information
  - Interop Assemblies to map .NET and COM identities
  - Managed code vs. Unmanaged code
    - COM Interop API

http://msdn2.microsoft.com/en-us/library/sd10k43k.aspx

Key Concepts

The Languages

- VBA
  - The built-in customization language of ArcGIS Desktop
  - Create User Interface (UI) Controls and Macros
- VB6 / .NET
  - Create Active X (COM) .dlls for ArcGIS Desktop and ArcGIS Engine applications (.exe)
  - Can extend ArcGIS System in many areas

Key Concepts

Entry Points for ArcGIS Customizations

VBA

- Use ‘ThisDocument’ and ‘Application’ variables as the gateway

VB6 and .NET

- Use ‘hook’ object passed into the OnCreate method to reference the applications (ArcMap, ArcCatalog)
- Use ‘HookHelper’ object when the command is to support a variety of hook objects

Demo

Language Differences

Data Type Differences

VBA/VB6	VB.NET	C#	CLR
Variant/Object	Object	object	System.Object
String	String	String	System.String
Boolean	Boolean	bool	System.Boolean
(n/a)	Char	char	System.Char
Single	Single	float	System.Single
Double	Double	double	System.Double
Currency	Decimal	decimal	System.Decimal
Date	Date	(n/a)	System.DateTime
(n/a)	(n/a)	sbyte	System.SByte
Byte	Byte	byte	System.Byte
Integer	Short	short	System.Int16
(n/a)	(n/a)	ushort	System.UInt16
Long	Integer	int	System.Int32
(n/a)	(n/a)	uint	System.UInt32
(n/a)	Long	long	System.Int64
(n/a)	(n/a)	ulong	System.UInt64

Language Differences

Inheritance and Polymorphism

- Use existing code to create new behavior
- VBA/VB6 uses only Implements (Polymorphism)
  http://msdn2.microsoft.com/en-us/library/7z6hzchx(VS.80).aspx
- .NET uses both Implements and Inherits (Inheritance)
  http://msdn2.microsoft.com/en-us/library/1yk8s7sk(VS.80).aspx

Language Differences

Inheritance and Polymorphism (continued)

- Implements (Polymorphism)
  - Must implement all the members declared in the interface
  - Can Implement multiple interfaces
- Inherits (Inheritance)
  - Take advantage of functionality of base classes
  - Can only Inherit one Class
  - ESRI.ArcGIS.ADF Assembly

Demo

Language Differences

Error Handling

- VBA/VB6
  - On Error GoTo / On Error Resume Next
  - http://www.cpearson.com/excel/ErrorHandling.htm

```
On Error GoTo ErrorHandler
 ...
 Exit Sub
ErrorHandler:
   MsgBox CStr(Err.Number) & " " &
  Err.Description
```

Language Differences

Error Handling (continued)

- .NET
  - Try Catch Finally / Throw an Exception
  - http://support.microsoft.com/kb/315965

```
Try
  'Code that may raise an error
   If x = 0 Then
     Throw New System.DivideByZeroException
   End If
Catch ex As Exception
  'Code to handle the error
  MessageBox.Show(ex.InnerException.ToString)
Finally
  'Code to do any final clean up
  'This sections always executes
End Try
```

Demo

Language Differences

Variable declaration and Instantiation

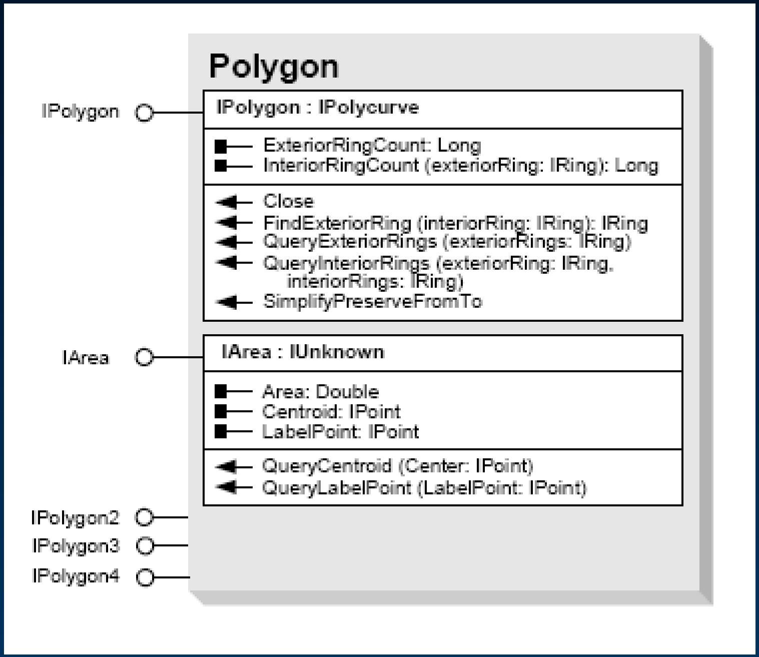
- VBA/VB6
  - Use Dim to declare a variable
  - Use Set to assign an object instance to a variable
- VB.NET
  - Set keyword is gone
  - Can perform the declaration & instantiation on one line

```
'VB.NET Two line version                  'VB6
Dim aPoint as IPoint                      Dim pPolygon as IPolygon
aPoint = New PointClass                  Set pPolygon = New Polygon
...
'VB.NET One line version
Dim aPolygon as IPolygon = new PolygonClass
```

Language Differences

Traversing COM Interfaces

- Switching between interfaces on a Class is central to traversing ArcObjects


Polygon
IPolygon
IPolygon : IPolycurve
ExteriorRingCount: Long
InteriorRingCount (exteriorRing: IRing): Long
Close
FindExteriorRing (interiorRing: IRing): IRing
QueryExteriorRings (exteriorRings: IRing)
QueryInteriorRings (exteriorRing: IRing, interiorRings: IRing)
SimplifyPreserveFromTo
IArea
IArea : IUnknown
Area: Double
Centroid: IPoint
LabelPoint: IPoint
QueryCentroid (Center: IPoint)
QueryLabelPoint (LabelPoint: IPoint)
IPolygon2
IPolygon3
IPolygon4

- In VBA/VB6 this is call Query Interface (QI)
- In .NET this is call Casting

Language Differences

Traversing COM Interfaces (continued)

- VBA/VB6 (Query Interface):

```
Dim areaPolygon as IArea
'CoCreate Object
Set areaPolygon = New Polygon
Dim geometry as IGeometry
If TypeOf geometry is IArea Then
  Set geometry = areaPolygon 'QI
End If
```

Language Differences

Traversing COM Interfaces (continued)

- VB.NET (Casting – There are 4 options):

```
Dim areaPolygon as IArea = New PolygonClass

‘ Analogous to QI in VB6 – Throws InvalidCastException on error
Dim geometryPolygon1 as IGeometry = areaPolygon

‘ Throws InvalidCastException on error
Dim geometryPolygon2 as IGeometry = CType(areaPolygon,IGeometry)

‘ Throws InvalidCastException on error
Dim geometryPolygon3 as IGeometry = DirectCast(areaPolygon, IGeometry)

‘ Returns Nothing on error
Dim geometryPolygon4 as IGeometry = TryCast(areaPolygon, IGeometry)
```

Language Differences

Traversing COM Interfaces (continued)

- A .NET alternative to eliminate Casting (or QI) altogether for creatable classes:

```
‘VB.NET
Dim aPolygon as New PolygonClass
```

Demo

Language Differences

Event Handlers in VBA/VB6

- Only supports one outbound interface per coclass
- Uses dummy coclass to access other outbound interfaces

```
'Default outbound interface
Private WithEvents m_pActiveViewEvents as Map
Set m_pActiveViewEvents = MapControl1.Map

'Other outbound interface
Private WithEvents m_pMapEvents as MapEvents
Set m_pMapEvents =MapControl1.Map
```

Event Handlers in .NET

- Uses Delegate to communicate
- Event interfaces have an _Event suffix
- Access multiple outbound interfaces
- Wire ArcObjects .NET events
  - Step 1: Cast the event interface
  - Step 2: Register the event handler method
  - Step 3: Implement the event handler method
  - Step 4: Unwire the event

Demo

Language Differences

Replace Win32 API calls with .NET Framework libraries

- File I/O functions resides in System.IO assembly

Win32 function	.NET Framework API
CopyFile	System.IO.File.Copy
CreateDirectory	System.IO.Directory.CreateDirectory
GetCurrentDirectory	System.IO.Directory.GetCurrentDirectory
ReadFile	System.IO.FileStream.Read
SearchPath	System.IO.File.Exists

http://msdn2.microsoft.com/en-us/library/aa302340.aspx

Language Differences

Replace Win32 API calls with .NET Framework libraries

- Printer-related functions reside in System.Drawing.Printing

Win32 function	.NET Framework API
EnumForms	System.Drawing.Printing.PrinterSettings.PaperSizes
EnumPrinters	System.Drawing.Printing.PrinterSettings.InstalledPrinters
GetDefaultPrinter	System.Drawing.Printing.PrinterSettings constructor System.Drawing.Printing.PrinterSettings.PrinterName
GetForm	System.Drawing.Printing.PrinterSettings.PaperSizes
GetPrinter	System.Drawing.Printing.PrinterSettings.PrinterName System.Drawing.Printing.PrinterSettings properties
SetPrinter	System.Drawing.Printing.PrinterSettings.PrinterName System.Drawing.Printing.PrinterSettings properties
DeviceCapabilities	System.Drawing.Printing.PrinterSettings properties

Language Differences

Replace Win32 API calls with .NET Framework libraries

- All GDI+ drawing code resides in System.Drawing.DLL

Win32 function	.NET Framework API
BitBlt	System.Drawing.Graphics.DrawImage
CreateBitmap	System.Drawing.Bitmap constructor
CreatePen	System.Drawing.Pen constructor
GetDC	System.Drawing.Graphics.GetHdc
ReleaseDC	System.Drawing.Graphics.ReleaseHdc
GetDeviceCaps	System.Drawing.Graphics properties System.Drawing.Printing.Printersettings
Rectangle	System.Drawing.Graphics.DrawRectangle

Demo

Language Differences

System.Runtime.InteropServices Namespace

- Not all Win32 functions have equivalent .NET Framework APIs
- Support COM Interop and platform invoke (PInvoke) services
  - DllImportAttribute Class
  - MarshalAsAttribute Class

http://pinvoke.net

```
‘VB.Net
<System.Runtime.InteropServices.DllImportAttribute("gdi32.dll")>
Private Shared Function DeleteObject (ByVal hObject As IntPtr) As Boolean
End Function

//C#
[System.Runtime.InteropServices.DllImport("gdi32.dll")]
public static extern bool DeleteObject(IntPtr hObject);
```

Language Differences

Using Controls embedded on a Windows Form

- IToolControl
  - ComboBox, EditBox, ImageList, ImageCombo

- VB6
  - Can access Windows Forms directly
  - Form Load event automatically executed

- .NET
  - Must create an instance of the Windows Form

Language Differences

Bitmap Property for Commands and Tools

- VB6
  - Returned type as an esriSystem.OLE_Handle
- .NET
  - Returned type as an Integer
  - For Polymorphic classes, use Bitmap.GetHbitmap function and clean up with gdi32.dll DeleteObject function
  - For Inheritance classes, set the m_bitmap property

Demo

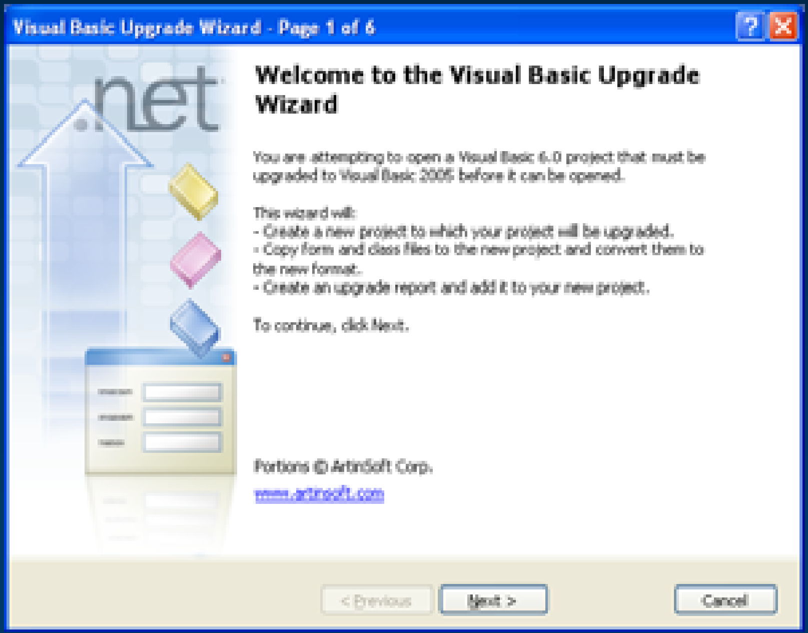
Visual Studio Migration Features

- Microsoft
  - Visual Basic Upgrade Wizard
- ESRI
  - ArcGIS Code Converter
  - ArcGIS Project Wizard Templates (choosing the right one)
  - Item Templates

Visual Studio Migration Features

Microsoft Visual Basic Upgrade Wizard

- Converts VB6 projects to VB.NET projects
- There are several limitations
  - Common dialogs
  - MsgBox
  - Win32API
  - .etc...



Welcome to the Visual Basic Upgrade Wizard

You are attempting to open a Visual Basic 6.0 project that must be upgraded to Visual Basic 2005 before it can be opened.

This wizard will:
- Create a new project to which your project will be upgraded.
- Copy form and class files to the new project and convert them to the new format.
- Create an upgrade report and add it to your new project.

To continue, click Next.

Portions © ArtinSoft Corp.
www.artinsoft.com

Visual Studio Migration Features

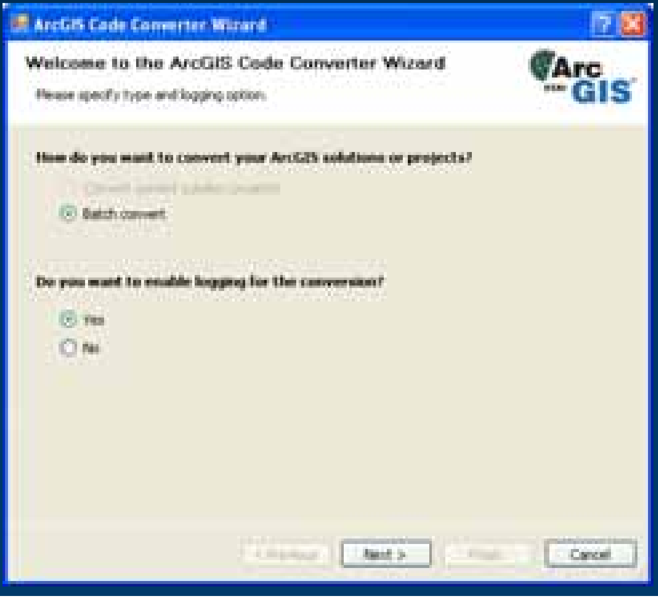
ArcGIS Code Converter

- Converts ArcGIS 8.3 to ArcGIS 9.x namespaces in the code

```
Dim pPoly As ESRI.ArcObjects.Core.IPolygon ‘8.3 namespace
Dim pPoly As ESRI.ArcGIS.Geometry.IPolygon ‘9.x namespace
```

- Appropriate project references will be updated

  `ESRI.ArcGIS.Utility` assembly is replaced with `ESRI.ArcGIS.ADF`



Welcome to the ArcGIS Code Converter Wizard

Please specify type and logging option.

How do you want to convert your ArcGIS solutions or projects?

Convert opened solution/projects [✓ Batch convert]

Do you want to enable logging for the conversion?

[✓ Yes] No

Visual Studio Migration Features

ArcGIS Project Wizard Templates (choosing the right one)

- **First, decide on the type of solution you need**
- **Then, choose the appropriate ArcGIS Project Template feature of the Visual Studio Integration Framework to use**

Out-Of-Process (.exe)
ArcGIS Engine Windows Application
ArcGIS Engine Console Application

In-Process (.dll)
ArcGIS Desktop Class Library General
ArcGIS Desktop Class Library ArcGlobe
ArcGIS Desktop Class Library ArcScene
ArcGIS Desktop Class Library ArcCatalog
ArcGIS Desktop Class Library ArcMap
ArcGIS Engine Class Library

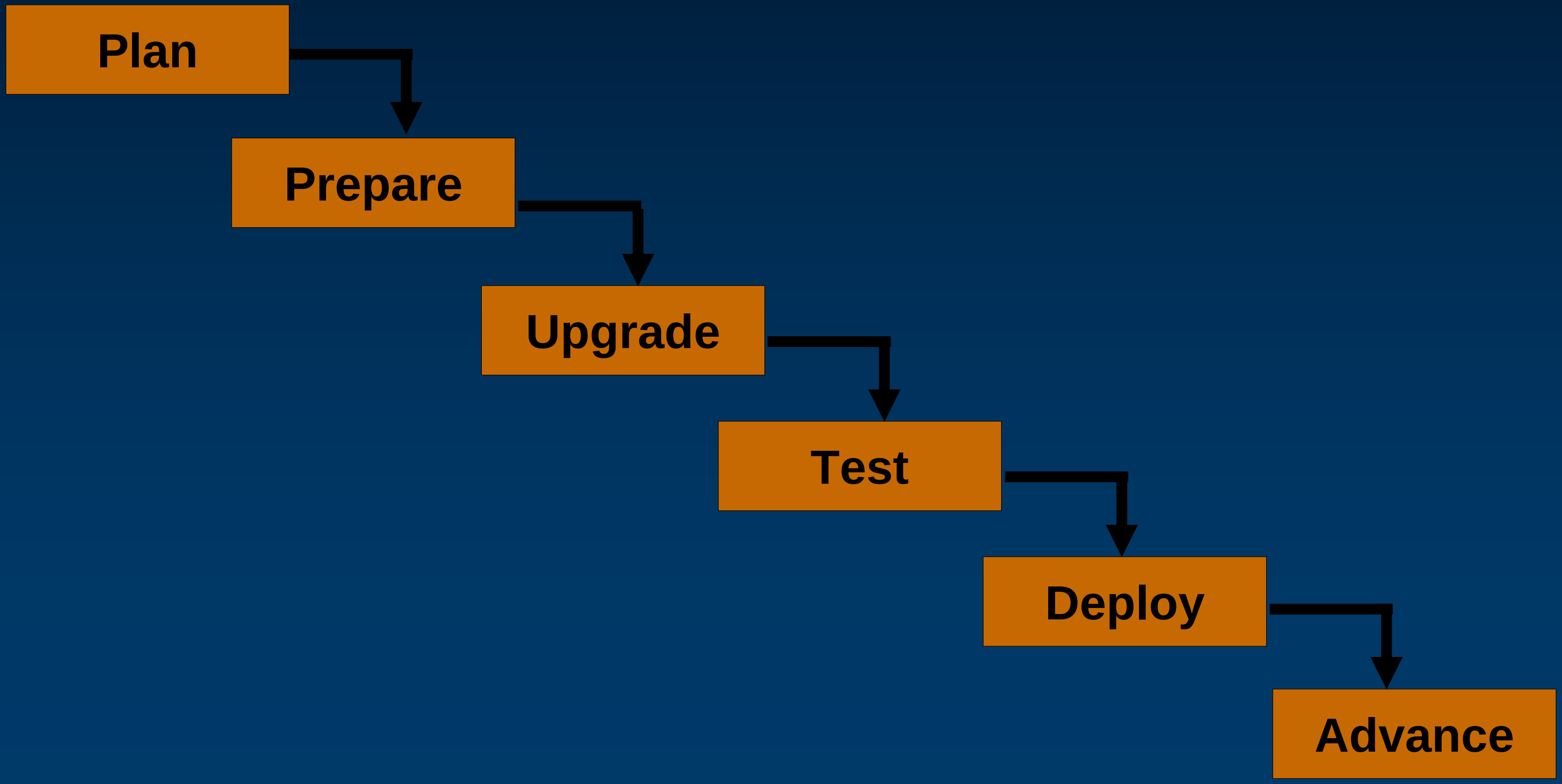
Visual Studio Migration Features

Item Templates

- Default code templates for adding ArcGIS customizations
  - Component category registration
  - Implemented interface stubs and common members
- Accessed by
  - Context Menu: Right click project name in the Solution Explorer > Add > New Item
  - Menus: Project > Add New Item

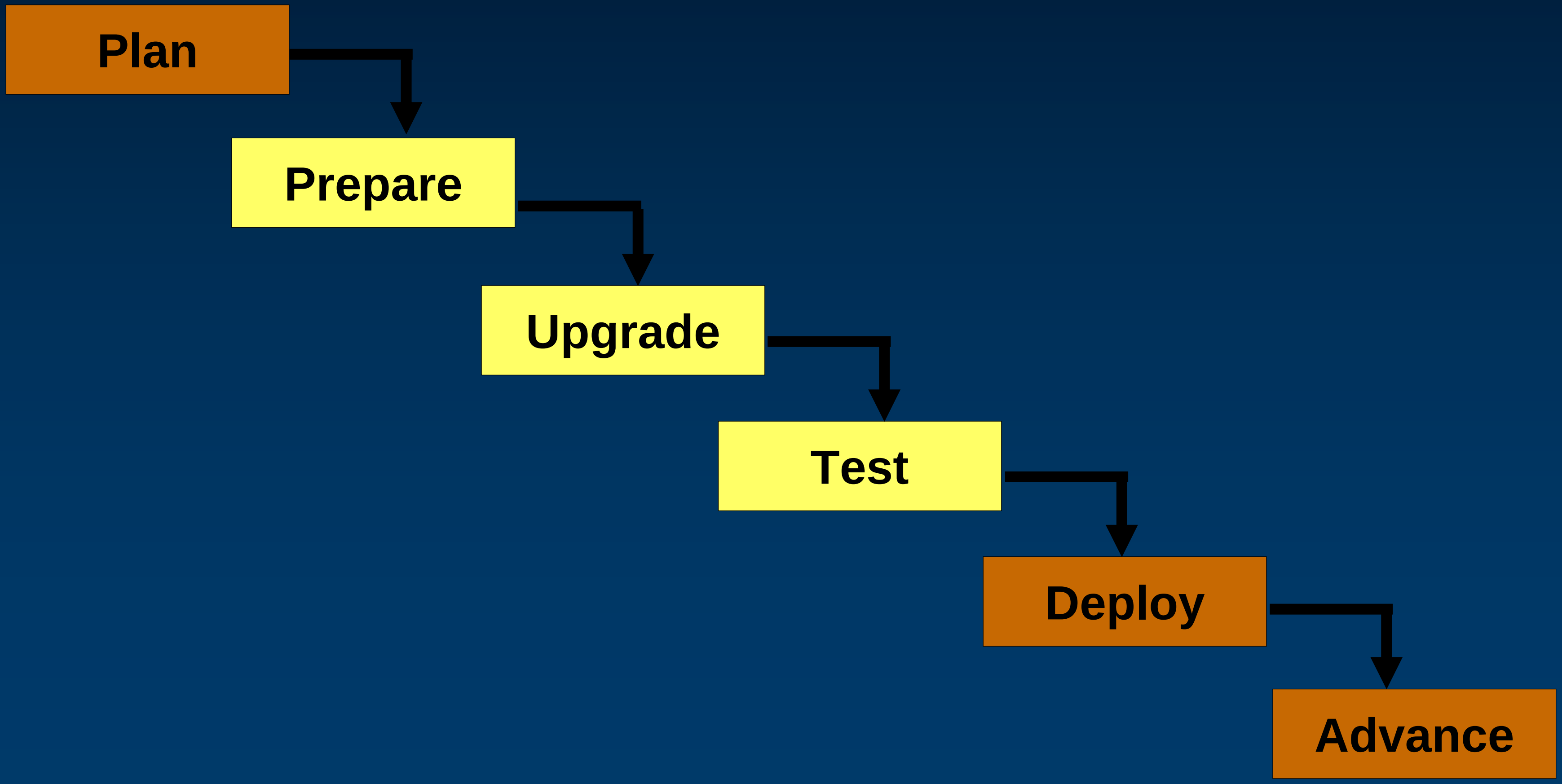
Bringing It All Together

- The upgrade process



Bringing It All Together

- A demo showing a VB6 to VB.NET project conversion



Bringing It All Together

Prepare

- Verify unused library references
- Modularize code as much as possible
  - Separate the business logic of the application from the Events
  - Parameterize functions where possible
  - Minimize the use of global variables
- "Option Strict" on, Option Explicit
  - Use early-bound data type, check the variable types if not possible
- Others:
  - Zero-bound arrays, avoid default properties

Bringing It All Together

Upgrade

- **Create a auto-translated VB.NET code project**
  - Use the Visual Studio code converter to import the VB6 project to .NET
  - If the ArcObjects code is in 8.x, use the ArcGIS .NET Code Converter
- **Create a new VB.NET code project to hold the final code**
  - Use ESRI’s Visual Studio IDE Integration features to create the Framework stubs
  - Copy the business logic from the auto-translated project
  - Edit the Framework stubs to use the business logic
  - Fix the language specific constructs and GUI forms/controls that did not auto-translate
  - Delete auto-translated comments

Bringing It All Together

Upgrade (continued)

- **Common issues:**
  - Compile Error/Warning
  - Wizard creates both class and interface definitions for user-defined interfaces
  - Use OLE- automation-compliant interfaces (IGeometryBridge)
  - COM registration
  - Component category registration
  - OLE_HANDLE values
  - Events

'<interface>__<member>' cannot implement '<member>'.
because there is no matching property on interface '<interface>'

```
Private ReadOnly Property ICommand_Bitmap() As Integer Implements ICommand.Bitmap
```

“Couldn't resolve default property of object <variable name>”

Demo

Bringing It All Together

Migrating a VBA application to .NET

- From the VBA Editor export the .bas, .cls, and .frm files to disk
- Create a new VB6 project and add the exported code files
- For each .frm create a matching .cls file and copy/paste code to it
  - The VBA forms (.frm) GUI will not translate to .NET, but you can salvage the code
- Add all of the ESRI References to the project
- Save the new VB6 project
- Follow the 'Migrating a VB6 application to .NET' recommendations

Additional Resources

Resources for Migration

- Microsoft's patterns and practices book: Upgrading Visual Basic 6.0 Applications to Visual Basic .NET and Visual Basic 2005
  http://www.microsoft.com/downloads/details.aspx?FamilyId=7C3FE0A9-CBED-485F-BFD5-847FB68F785D&displaylang=en
- ESRI's EDN document series: Programming with .NET
  http://edndoc.esri.com/arcobjects/9.3/NET/12BF6F08-1B25-4002-9640-73D4EB0E6CED.htm
- ESRI's EDN document: Migrating and upgrading your code
  http://edndoc.esri.com/arcobjects/9.3/NET/13ef8415-8314-4bc9-9b77-4b03599c2c11.htm

Additional Resources

C# Advice

- get_ and set_

```
'VB.NET
Dim layer As ILayer = map.Layer(0)
//C#
//ILayer layer = map.Layer(0);
ILayer layer = map.get_Layer(0);
```

- A good VB.NET to C# converter is available by Tangible Software Solutions (http://www.tangiblesoftwaresolutions.com/)
- Casting
- Events wiring
- Namespaces
- Using System.Type.Missing to indicate optional or undefined parameters

```
'VB NET
IAvtiveView.PartialRefresh(esriViewDrawPhase.esriViewGraphics)
//C#
IActiveView.PartialRefresh (esriViewDrawPhase.esriViewGraphics,
   Type.Missing, Type.Missing);
```

Additional Resources

Deployment

- Use regasm.exe to register .NET Assemblies
- Component categories done automatically with .NET project
  http://edndoc.esri.com/arcobjects/9.2/NET/33903846-919f-48b4-a4b1-ef97680ddb7b.htm
- Tech session: “Deploying Desktop Application”
  – Tuesday 4:30pm Primrose rooms C and D
  – Wednesday 1:00pm Pasadena/Sierra/Ventura

Additional Resources

Other recommended sessions

- Extending the ArcGIS Desktop Applications
  - Wednesday 10:30am Catalina/Madera
  - Wednesday 2:45pm Smoketree rooms A to E
- Developing .NET Applications for ArcGIS Engine
  - Tuesday 2:45pm Primrose rooms C and D

In Conclusion…

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