.NET Experiences

Chris Manhard and David Hearn

OSI Software, Inc.

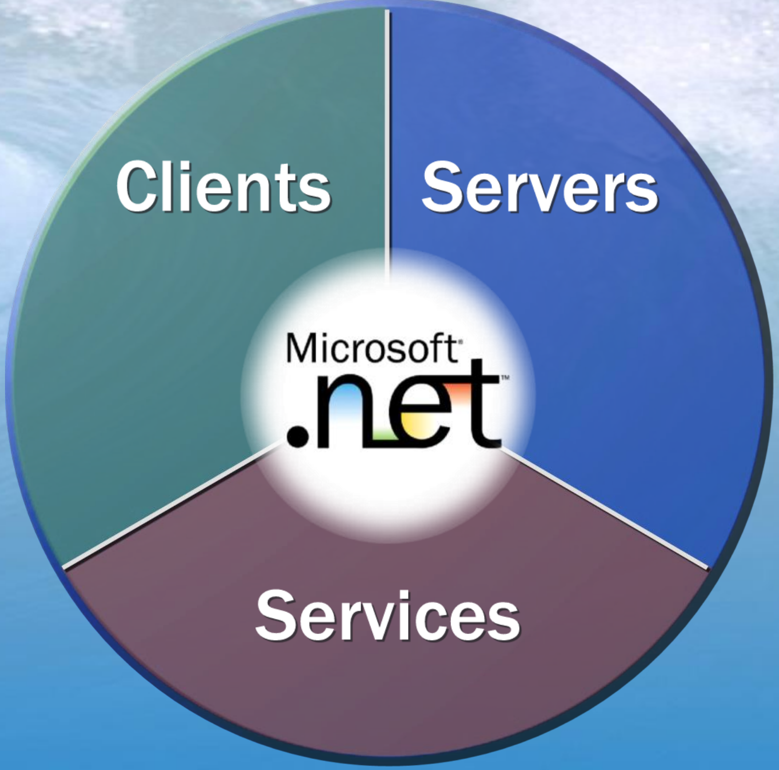
Agenda

- Microsoft .NET overview
- PI Application Framework and .NET
- Development Experiences
- Using .NET with existing PI tools
- Opinions and Recommendations

Microsoft’s .NET Goals

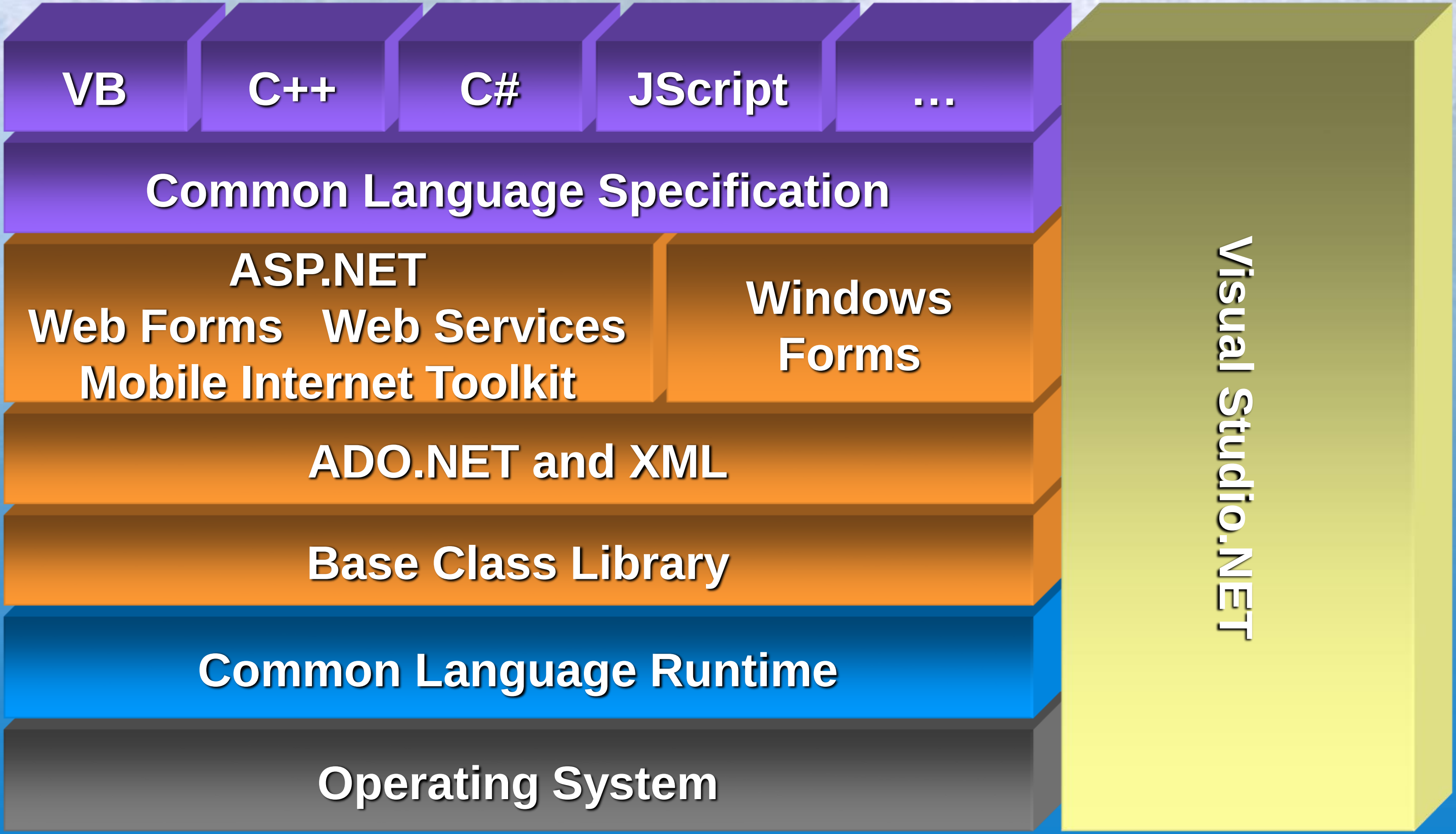
- Unifies programming models
- Dramatically simplifies development
- Supports multiple programming languages
- Provides robust execution environment
- Natively supports XML Web Services

Microsoft .NET Platform



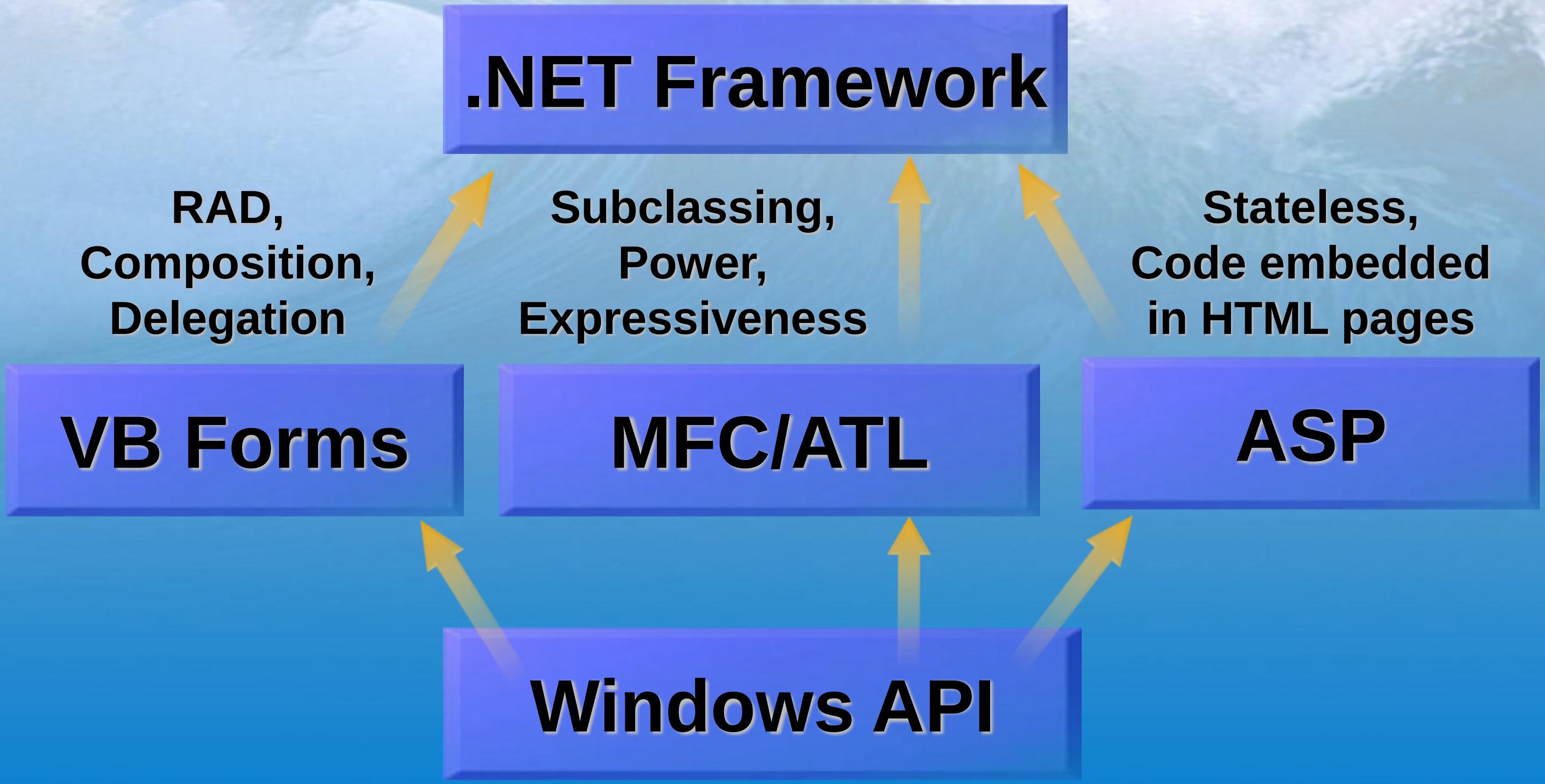
- Consistent programming model
- Includes clients, servers, services
- Development tools

Microsoft .NET Architecture

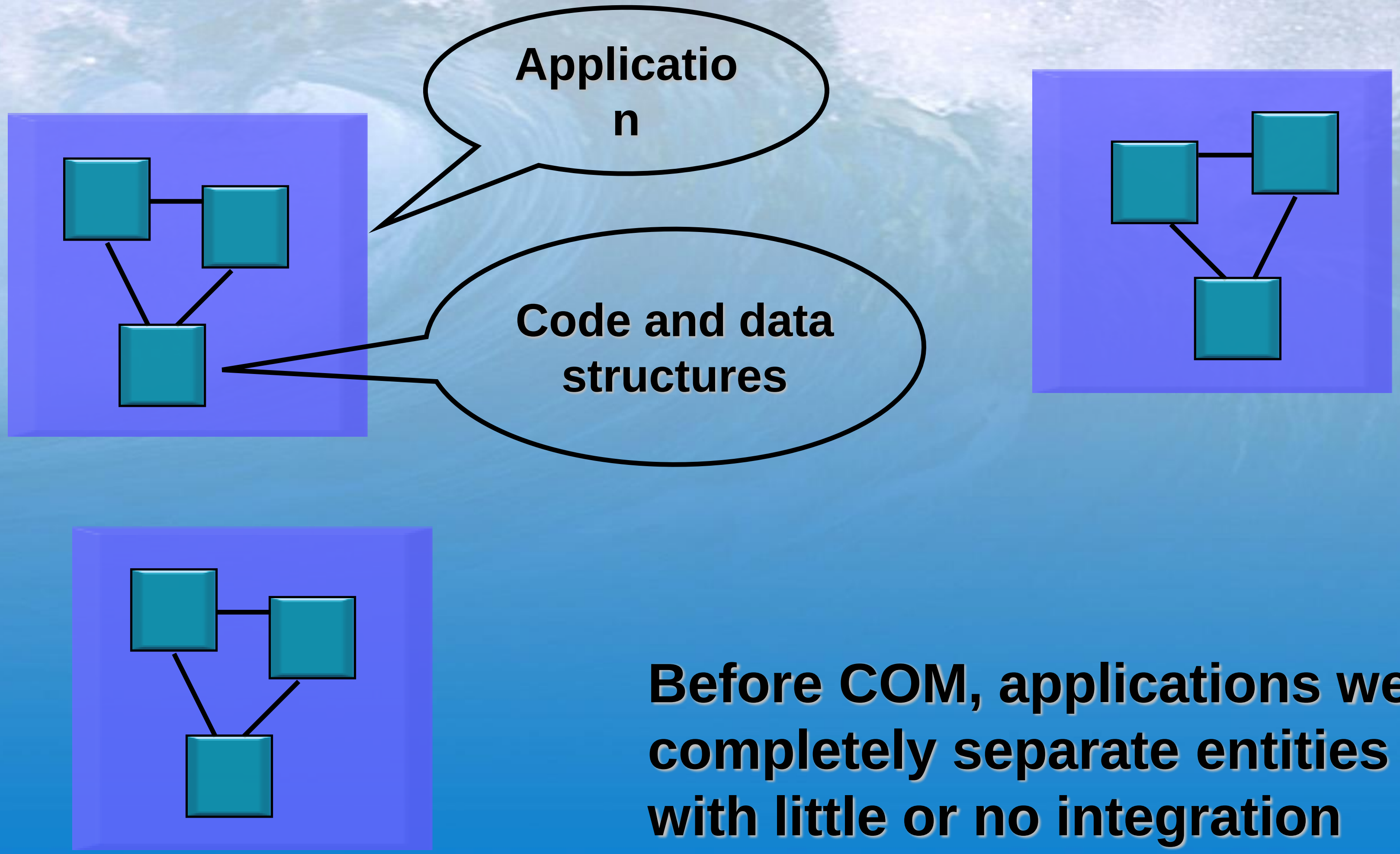


Unify Programming Models

Consistent API availability regardless of language and programming model

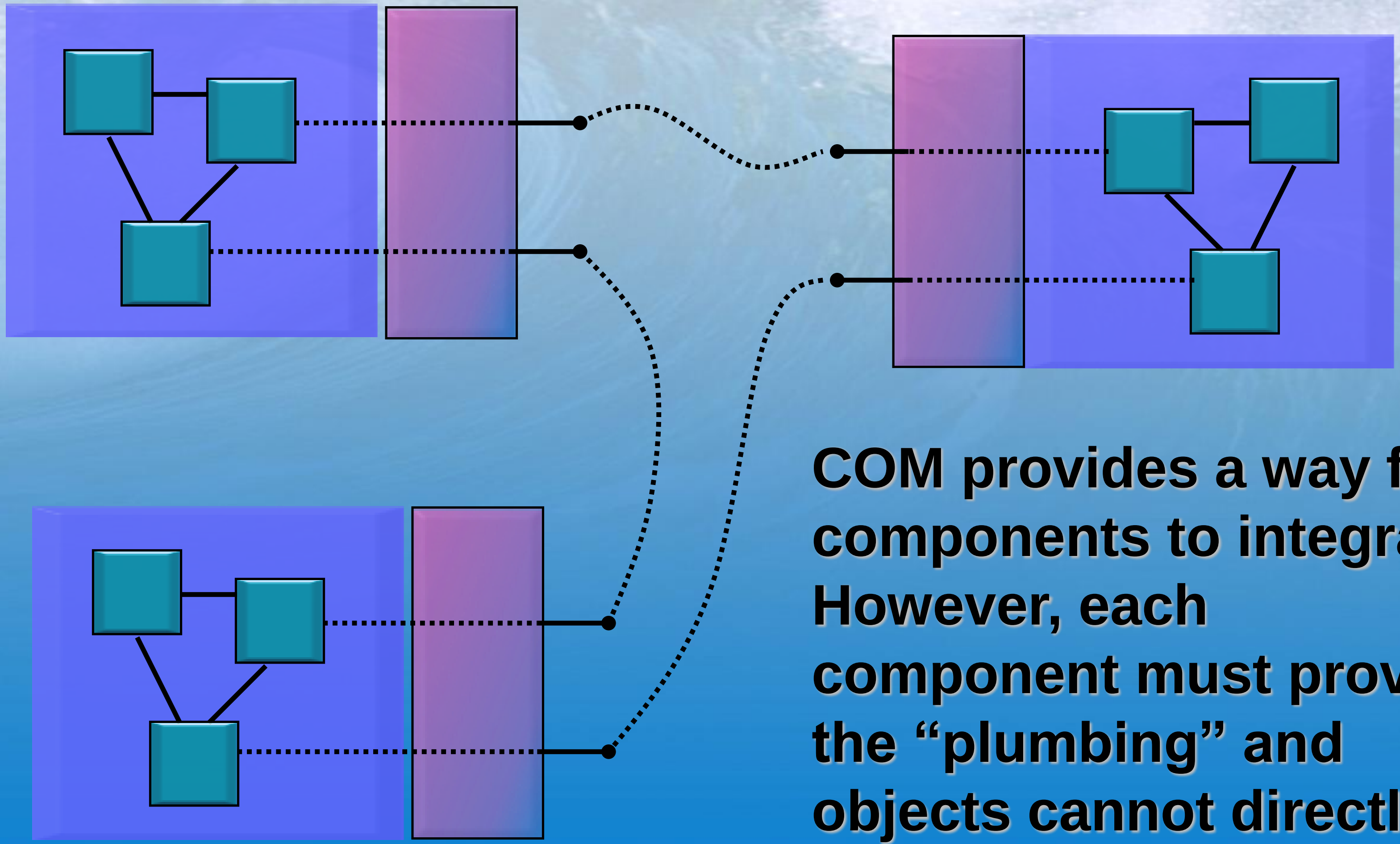

.NET Framework
RAD, Composition, Delegation
Subclassing, Power, Expressiveness
Stateless, Code embedded in HTML pages
VB Forms
MFC/ATL
ASP
Windows API

The .NET Evolution – Before COM



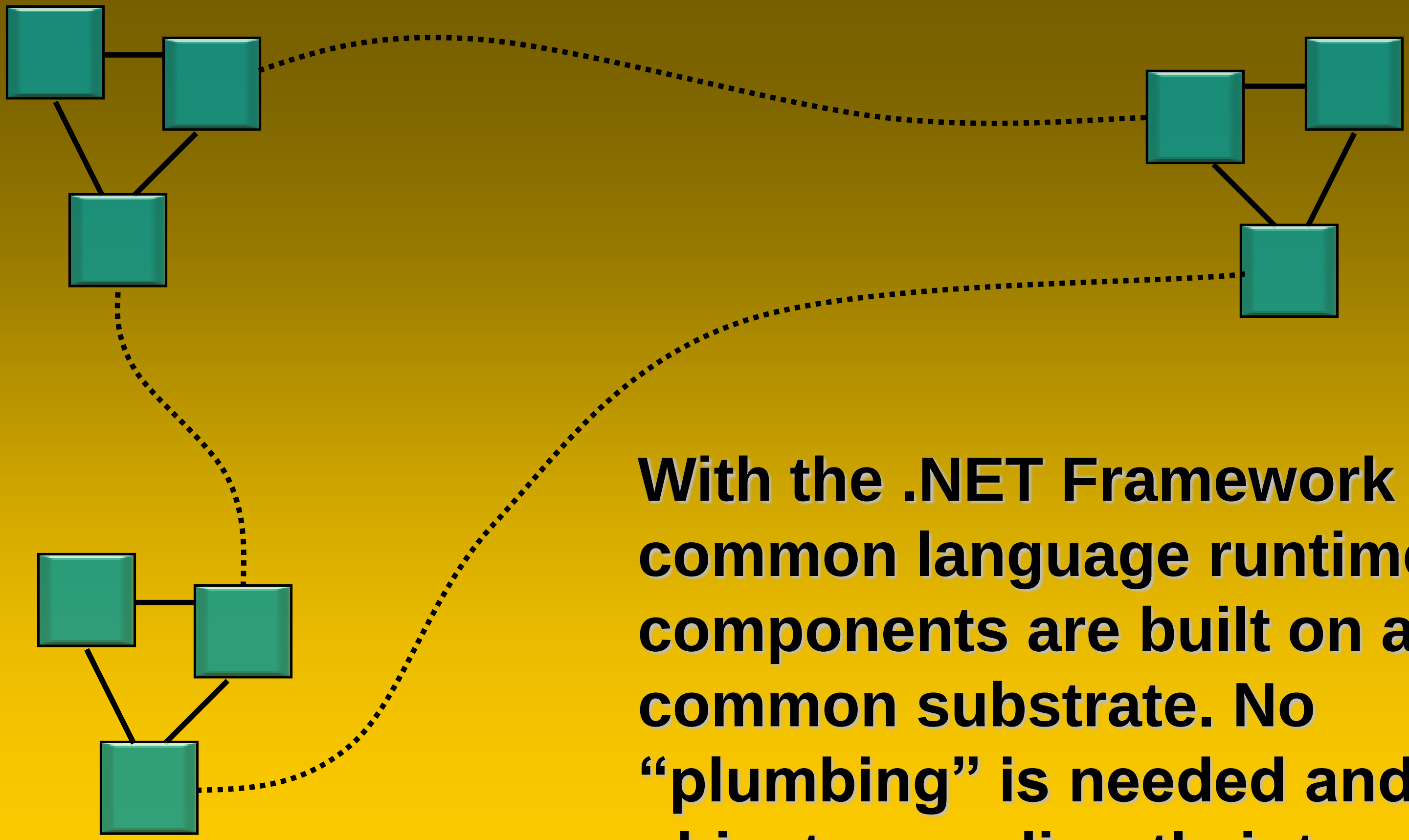
Before COM, applications were completely separate entities with little or no integration

The .NET Evolution - COM



COM provides a way for components to integrate. However, each component must provide the "plumbing" and objects cannot directly interact.

The .NET Evolution - Now



With the .NET Framework common language runtime, components are built on a common substrate. No “plumbing” is needed and objects can directly interact

Robust Environment

- Automatic lifetime management
  - All objects are garbage collected
- Exception handling
  - Error handling 1st class and mandatory
- Type-safety
  - No buffer overruns, unsafe casts, or uninitialized variables
  - Base types are treated as true objects
- Deployment and management
  - Assemblies, side-by-side execution
  - No more DLL hell!

Assemblies

- Unit of deployment
  - One or more files, independent of packaging
  - Self-describing via manifest
- Versioning
  - Provided by compiler use of attributes
  - Policy per-application as well as per-machine
- Security boundary
  - Assemblies are granted permissions
  - Methods can demand proof that a permission has been granted to entire call chain
- Types named relative to assembly
- Shared assemblies placed in GAC

PI Data from a Web Service Demo

- What is a Web Service
- Creating a web service
- Demo using PI OLE-DB
- Other ways to create a service

PI Application Framework Background

- Developing PI Application Framework
- Business Logic Layer
- Distributed Architecture
- Desire to publish / consume data across the various channels

PI Application Framework C++ Development

- Used C++ and ATL template library
- Common code abstracted to templates
- Override default template methods

PI Application Framework C# Development

- Direct access to objects
- Methods declared as ‘internal’ or ‘public’
- Single inheritance of base class
- Reduction in code
- Improved error handling
- Faster compile times

Reduction in Code – iterating a COM object using C++

```
// Create the PISDK object
PISDK::IPISDKPtr spPISDK;
spPISDK.CreateInstance("PISDK.PISDK");
PISDK::ServersPtr spPIServers= spPISDK->Servers;

// Enumerate over the collection using sequential access
HRESULT hr;
IEnumVARIANTPtr spEnum= spPIServers->Get_NewEnum();
do
{
    ULONG numFetched;
    CComVariant varItem;
    hr= spEnum->Next(1, &varItem, &numFetched);
    if(hr == S_OK)
    {
        PISDK::ServerPtr spPIServer(varItem);
        MessageBox(NULL, spPIServer->Name, "PI Server", MB_OK);
    }
}
while(hr == S_OK);
```

Reduction in Code – iterating a COM object using C#

```
// Create the PISDK object
PISDK.IPISDK piSDK= new PISDK.PISDKClass();

// Enumerate over the collection using sequential access
foreach(PISDK.Server item in piSDK.Servers)
{
     MessageBox.Show(item.Name, "PI Server");
}
```

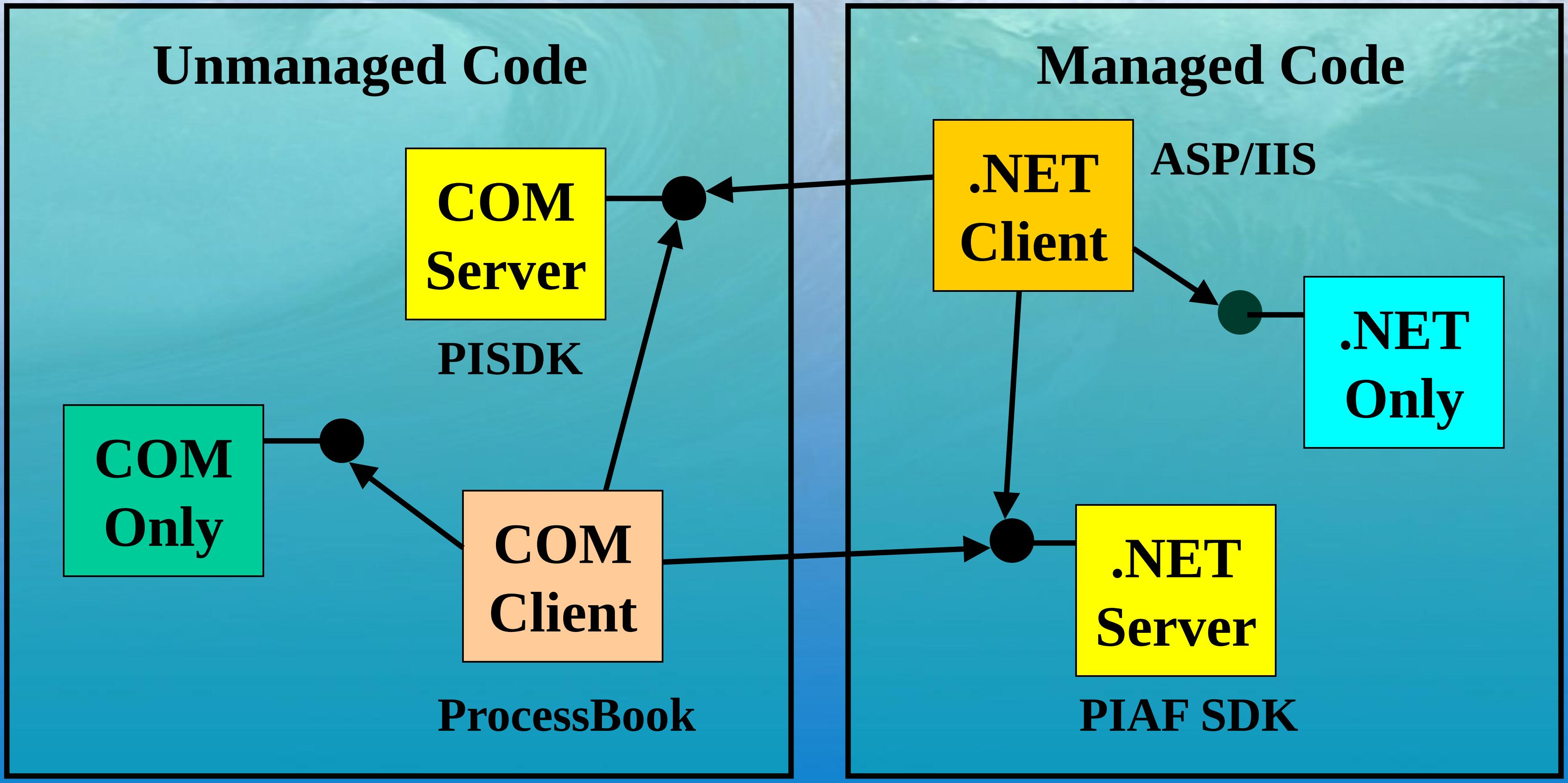
Error Handling in C++

```
try
{   // Code which might throw an exception.
    PISDK::ServerPtr spPIServer= spPIServers->GetDefaultServer();
    if( spPIServer == NULL )
        return AtlReportError(CLSID_PIAF,
                    “Failed to get default server”,  IID_Server, hres);
}catch(…)
{
    return AtlReportError(CLSID_PIAF,
                    “Failed to get default server”);
}
```

Error Handling in C#

```
PISDK.Server piServer= spPIServers.DefaultServer;
    if( piServer == null)
        return System.InvalidOperation(“Failed to get default server”);
```

Interop With COM



Invoking PI-SDK from .NET

- Use References, just as in VB6
- Use PISDK 1.2 if calling from IIS
- Some parameters must be ‘boxed’ in C#

```
namedValues.Add(“Level”, 0.0)
```

becomes

```
Object val = 0.0;
namedValues.Add(“Level”, ref val);
```

- Must create top level PISDK object
- PISDK.PISDK and PISDK.Server conflicts

Web Form Demo with PISDK

<This page is intentionally blank>

Calling the PI API from .NET

1. Must use locking if calling from within IIS
2. Run Migration Utility on PIAPI32.BAS
3. Add Structure Layout Attributes

   <struct layout sequential>
4. Change ANY types to defined types
5. Change buffers from Strings to StringBuilder
6. Rename variable as appropriate
7. Arrays should marshal fine

Hints and Tips

- Learn the .NET framework
- Option Strict On in VB.NET
- Use StringBuilder instead of String
- Don’t modify a collection while iterating it
- Add ASPNET user account bug
- Use ComSourceInterfaces attribute with ‘typeof’ parameter when exposing events.
  [ComSourceInterfaces(typeof(OSIsoft.AFSDK._IAFCollectionEvents))]

Top 10 Favorite Things in .NET

- C# is easy transition
- Remoting capabilities
- ASP.NET simplifies web development
- Rich Framework
- Exception Handling
- Com Interop / PI
- Code Editor
- Development Environment
- Compile Speed
- Unified Development

Top 10 Least Favorite Things in .NET

- Information overload
- WinForms is step back from VC6
- All tools not yet integrated
- VB Migration Tool
- Poor support of VS6 resource editor
- Versioning – Side by Side hell?
- Install – not as easy as advertised
- Beta Software
- Can’t Write ActiveX Controls
- Change

When to use Microsoft .NET

- Building Distributed Applications (n-tier)
  - Especially client and business tiers
- Creating Web Services
- Creating ASP Web Site
- Creating small to medium size applications

When NOT to use Microsoft .NET

- Large Traditional Applications
- Smaller Applications, especially if distributing .NET Framework (21 Meg) is an issue, or Win95 support required.
- Creating ActiveX Controls
- Real-Time Requirements
- External Toolset Requirements
- Projects where migration might be difficult

Summary

What is .NET?

- Related Presentations
  - PI Application Framework, (351) Wed. 8:00 AM
  - Application Module Example Using the Application Framework (112) Mon. 1:50 PM, (352) Wed. 8:50 AM
  - Sigmafine 4.0 (342) Wed. 8:50 AM
- Demo Room (Tuesday, 1:00 pm to 6:00 PM)
  - PI Application Framework
  - Application Framework Applications
  - Sigmafine 4.0
- Questions