A Programmer’s view on .NET
Three Pillars of .NET

- Many devices/Languages
- XML Web Services
- Simpler programming model
The Marketeer’s Toolbox

- Product
- Wrench
- Price
- Axe
- Competition
- Hammer
The Programmer’s Toolbox

C# Bricks

COBOL Pictures

Perl Duct Tape

Haskell Super Glue
dumpbin /disasm user32.dll
Building a Compiler

Haskell

Libraries

Runtime

Intel

Libraries

Runtime

PPC

Libraries

Runtime

6502
Building a Compiler

- Intermediate Language
- Haskell
- Libraries
- Runtime
- Intel
- PPC
- 6502
VisualStudio.net

- "Drag and Drop" Web services development
- Fully supports .NET Framework
- Multi-language support
- Open, extensible development environment
- Easy to plug in new languages (Babel)
Three Pillars of .NET

- Many Devices/Languages
- Web Services
- Simpler Programming Model
CORBA and DCOM are not compatible.

CORBA and DCOM are connection oriented, doesn’t scale well.

CORBA implementations are not compatible.

location transparency does not hold on the Web.
HTTP protocol

ASP

<html>
...</html>

???????
XML Webservices
Enterprise Servers

- Functionality for development and management of .NET services:
  - Application Center 2000:
    - Reliability and scalability
  - Internet Security & Acceleration Server:
    - Firewall and proxy server
  - Commerce Server:
    - Analyzing site usage
  - BizTalk Server 2000:
    - Business process orchestration and business-to-business document interchange using XML
Enterprise Servers

- SQL Server 2000:
  - Easy-to-use database systems with native Extensible Markup Language (XML) support
- Host Integration Server 2000:
  - Integration with host systems and their data
- Mobile Information 2001 Server:
  - Integration with mobile devices
Three Pillars of .NET

- Many Devices/Languages
- Web Services
- Simpler Programming Model
void _OldSysKind();
short MajorVersion();
short MinorVersion();
short AttributeMask();
short AttributeStrings([out] SAFEARRAY(BSTR)* AttributeArray);
CoClasses* CoClasses();
Interfaces* Interfaces();
Constants* Constants();
Declarations* Declarations();
short TypeInfoCount();
void _OldGetTypeKind();
TypeInfo* GetTypeInfo([in] VARIANT* Index);
short GetTypeInfoNumber([in] BSTR Name);
VARIANT_BOOL IsSameLibrary([in] TypeLibInfo* CheckLib);

....
```cpp
#import "Bar.dll"
main (){ 
  IFoo *b;
  CoInitialize(NULL);
  CoCreateInstance (__uuidof(Bar), NULL, CLSCTX_SERVER, __uuidof(IFoo), (void **) &b);  
  .................. 
  ..................
  ..................
  ..................
  .................
}
```
using Bar;

void Main (){
    IFoo b = new Bar();
    ...........
    ...........
    ...........
    ...........
}

Don’t Try This at Home!
New App

X.dll
Version 2000
Replacing Newer Version with Older Version

An old bug is reintroduced, or new functions are now unavailable.
Replacing Older Version with Newer Version

New App

Old App

Implementation relied on “feature” removed in new version.

X.dll Version 2001

Other App
When a new DLL installed, W2K checks whether the changed DLL was protected.

If the new DLL doesn't have a valid signature, W2K restores the original DLL from `winnt\system32\dllcache`. 
# Side-By-Side Versioning

![Side-By-Side Versioning](image)

<table>
<thead>
<tr>
<th>Global Assembly Name</th>
<th>Version</th>
<th>Locale</th>
<th>Originator</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalendarGenerator</td>
<td>0.0.0.0</td>
<td></td>
<td>260c7b3d70dad570</td>
</tr>
<tr>
<td>gac</td>
<td>0.0.0.0</td>
<td></td>
<td>0f859bad23328929</td>
</tr>
<tr>
<td>gac</td>
<td>0.0.0.0</td>
<td></td>
<td>3b85ea9d475b7983</td>
</tr>
<tr>
<td>gac</td>
<td>0.0.0.0</td>
<td></td>
<td>56800ff85fea508e</td>
</tr>
<tr>
<td>gac</td>
<td>0.0.0.0</td>
<td></td>
<td>a6a1153729114360</td>
</tr>
<tr>
<td>mondrian</td>
<td>0.0.0.0</td>
<td></td>
<td>841b02f0baeddbcf</td>
</tr>
<tr>
<td>MultiGen</td>
<td>1.0.0.0</td>
<td></td>
<td>dc8a95b697cdce3e</td>
</tr>
</tbody>
</table>

7 object(s)
Before COM/DLLs, applications were completely separate entities with little or no integration.
COM provides a way for components to integrate. However, each component must provide the “plumbing,” and objects cannot directly interact.
Type Incompatibilities

Visual Basic String

C++ char array
No Reuse of Implementation

You have to encode “implementation inheritance” explicitly using aggregation.
Concurrency control is not built-in (that is, coclasses do not have a classwide lock).

Instead it has several sorts of “apartments”: MTA, STA, FTA
With the .NET Framework common language runtime, components are built on a common substrate. No "plumbing" is needed, and objects can directly interact.
using System;

namespace VirtualDog {
	public class Dog {
		public virtual void RollOver () {
		Console.WriteLine("Scratch my tummy.");
		Bark();
	}

		public virtual void Bark () {
		Console.WriteLine("WOOF WOOF (Dog)");
	}
}
}
Imports System
Namespace VirtualDog
    Public Class Mopje : Inherits Dog
        Public overrides Sub Bark ()
            Console.WriteLine("WOEF WOEF (Mopje)")
        End Sub
    End Class
End Namespace
import VirtualDog;

var d = new Dog();
var m = new Mopje();

d.RollOver();
m.RollOver();
Business model will change to include selling services and subscriptions and supporting non-Windows platforms.
.NET Advantages

- Thanks to the CLI we have:
  - Cross-language implementation inheritance
  - Cross-language exception handling
  - Cross-language threading
  - Cross-language debugging
  - ASP.Net pages in any language
  - XML Webservices in any language

- Third Party Languages include:
  - COBOL, Pascal, Oberon, APL, Fortran, …
  - Mondrian, Haskell, Mercury, Eiffel, …