

Course Structure

- (1) Sorting the field
 1. Goals, Basic Notions, History
 2. Overview of approaches
- (2) **Classic/Commercial Component Systems**
 1. Problems they solve, leave unsolved
 2. Corba, DCom, (EJB,) Web Services, **MS.Net**
 3. Comparison
- (3) Advanced Composition Concepts
 1. Architectural languages
 2. Aspect-oriented programming
 3. Generative programming
 4. Meta-programming/Invasive Composition
- (4) Problem Discussions

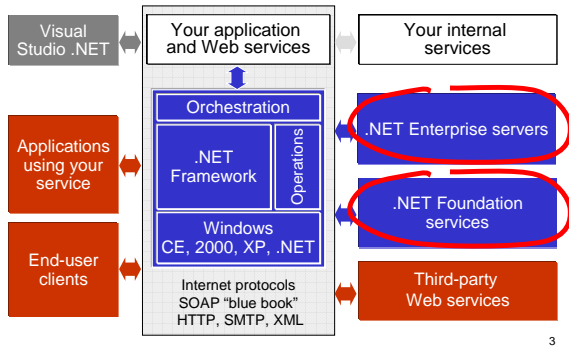
1

Disclaimer

- What follows is a sparse (and less fancy) version of presentations available from Microsoft Developer Network (MSDN)
- More information:
 - <http://msdn.microsoft.com>
 - <http://www.microsoft.com/net>
 - <http://msdn.microsoft.com/xml>
 - msnews.microsoft.com
 - microsoft.public.dotnet.general
 - microsoft.public.dotnet.xml

2

The .NET Platform



3

.Net Enterprise Servers

- Basic and general services
- Include:
 - Application Center
 - BizTalk Server
 - SQL Server
 - Commerce Server
 - Exchange Server
 - Host Integration Server
 - Internet Security and Acceleration Server
 - Mobile Information Server
- Web services based communication

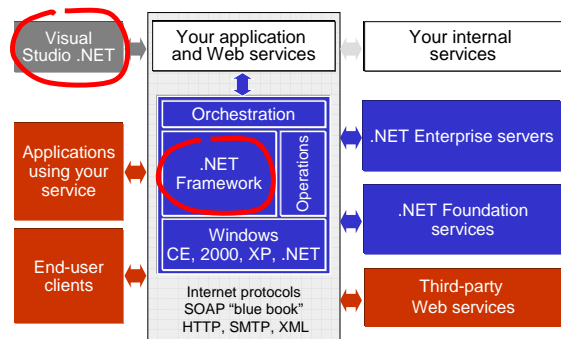
4

.Net Foundation Services

- Further general services
- MS Passport
- Storage .NET
- Maintaining private data is critical
 - Acceptance relays on
 - Security (objective)
 - Trust (subjective)
 - Large customers as reference
 - MSN portal Hotmail (bought, cracked!)
 - Passport (bought)
 - Privacy initiatives
 - TRUSTe Codex of fair information practices
 - No adds, no mining, no transmission

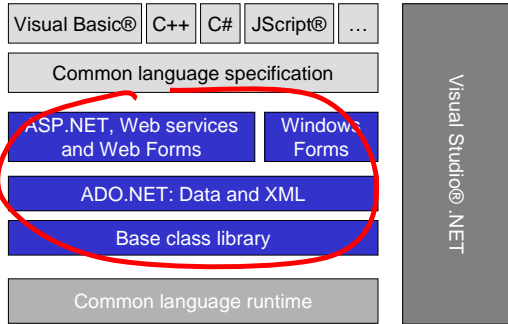
5

The .NET Platform



6

.NET Framework/Visual Studio .NET



7

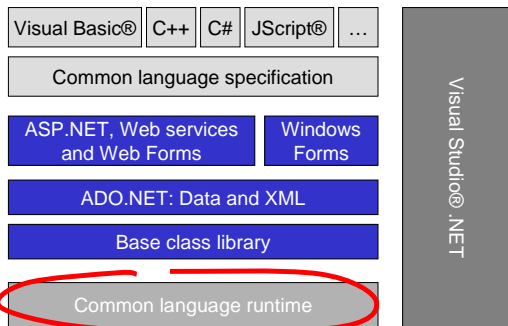
.NET Framework Services

Basic services, libraries

- ASP.NET: evolution of Active Server Pages (compiled)
- Web Forms: thin clients, integration with ASP.NET
- Windows Forms: rich clients
- ADO.NET
 - Serialization of objects to XML and back
 - evolution of ActiveX Data Objects
- XML support throughout

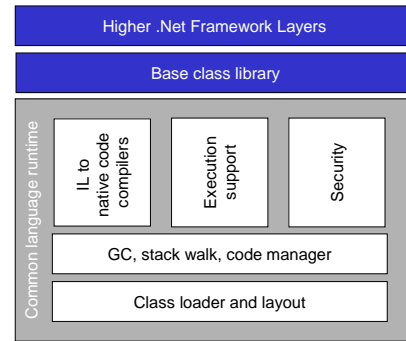
8

.NET Framework/Visual Studio .NET



9

CLR Architecture



10

CLR Goal

- Development
 - standard class framework (language independent)
 - automatic memory management (no reference counting)
 - mixed-language applications (incl. common type system, cross language inheritance, common meta-data, consistent error handling)
 - multiple platforms (due to JIT)
 - safe(r) execution
- Deployment
 - Components (assemblies) - uniquely identified, deployed together with the application
 - removal of registration dependency
 - safety: fewer versioning problems
 - the end of "DLL Hell"

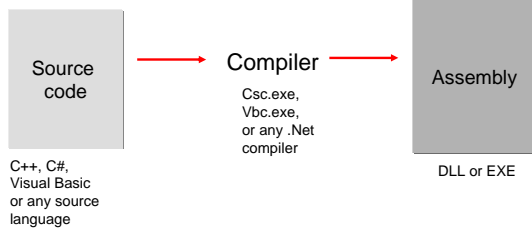
11

CLS and CLR

- Common Language Specification (CLS)
- Defines a Common Type System (CTS)
- Intermediate languages for any higher programming language
- Language transparency
 - on intermediate level by JIT compiling in the CLR (managed code)
 - on binary level as before in COM (unmanaged code)

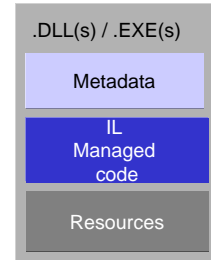
12

Compilation



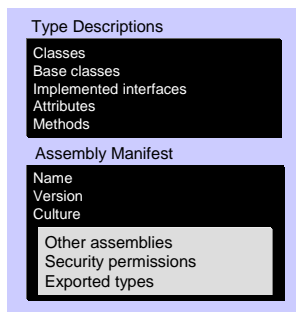
13

Assemblies



14

Metadata



15

Type Information

- Complete interface information (IDL)
- Automatically bound into assembly
 - inseparable
 - stored in binary format
- Describes every class type
- Can be used by Microsoft tools
 - automatic completion, e.g. by Visual Studio.NET,
- CLR: dynamic type safety, i.e., type test on access
- “Assemblies are self-documenting”

16

Assembly manifest

Used by code manager:

- **Name** – of the assembly, used in assembly resolution; same as the assembly file name minus the extension.
- **Version** – used by runtime; three sub-values: major and minor version number, a revision, and build information.
- **Culture** – allows correct assembly to be used or correct resources for assemblies with different resources for different cultures.
- **Other Assemblies** – information regarding what other assemblies are required by this assembly; list DLLs that need to be imported.
- **Exported Types** – references to assembly contained type implementation
- **Strong Name** – a public key from the publisher if the assembly has been given a strong name, adds a public key encryption to the file containing the assembly manifest.

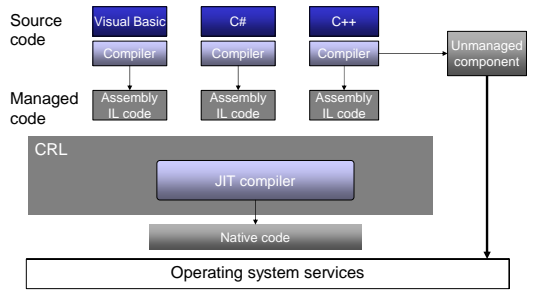
17

Application

- One or more assemblies
- Assemblies resolution using metadata
 - local (preferred)
 - Global Assembly Cache (GAC), registration using a special MS tool (Gacutil.exe)
- Different applications may use different versions of an assembly
 - easier software updates
 - easier software removal
- Replaces registry search (if succeeds locally)

18

Execution



19

My (!) .Net Conclusion

- Microsoft's new component technology
- Classic component system
- We discussed language transparency
 - CLR, CLS replace (D)COM's binary standard
 - Compatibility to COM (unmanaged code)
- Platform transparency: Mono, Open Source Implementation of .Net works fine
- Remote transparency
 - Remote access: Web Services as basic technology
 - Transparency: due to IDE tools support (VisualStudio.Net) hiding the differences between local and remote references

20

Course Structure

- (1) Sorting the field
 1. Goals, Basic Notions, History
 2. Overview of approaches
- (2) Classic/Commercial Component Systems
 1. Problems they solve, leave unsolved
 2. Corba, DCom, EJB, Web Services, MS.Net
 3. Comparison
- (3) Advanced Composition Concepts
 1. Architectural languages
 2. Aspect-oriented programming
 3. Generative programming
 4. Meta-programming/Invasive Composition
- (4) Problem Discussions

21