Microsoft .NET

A radical new approach to software usage, deployment and development?

Leon van Snippenberg

Atos Origin / IPS

e-mail: Leon.vanSnippenberg@atosorigin.com

What am I going to tell You?

- Why a “New” initiative?
- What is .NET?
- Who are the players?
- What does it mean for you?
- .NET as a SW development platform
- Roadmap
- Some final thoughts

Why a "New" initiative?

- The internet everywhere
  - Move to interconnected systems, SW as a service
- Hardware breakthroughs
  - Broadband, Wireless, smartcards, more CPU cycles/sec
- Solve incompatibilities
  - ASP, XML, DCOM, RMI etc
- Offer a friendlier programming environment
  - Type safety, garbage collection, "binary" portability, common virtual machine, explicit interfaces etc.

What is .NET?

- New platform for software development, deployment and usage
- Component based, successor of (D)COM
- Usable for building internet services, embedded software for hardware devices and everything in-between
- Offers standards like SOAP for distributed software to cooperate, based on omnipresent standards like HTTP and XML
- Offers tooling and languages (C, C++, C#...)

Who are the players?

- Microsoft has not monopolized .NET
  - SOAP is developed together with *IBM* and *Lotus* and handed over to IETF
  - C# and CLI (Common Language Infrastructure) will be standardized by ECMA. Proposed by *Fujitsu*, *HP*, *Intel* and *Microsoft*. Worked on by *IBM*, *Netscape*, *Sun*, *SHARE* and *Pixo*
  - Languages are developed by a large number of manufacturers, *Rational* is doing a Java implementation
  - *Corel* is expected to offer a Linux version

What does it mean for You?

- As an end user you:
  - Will control when, where and how your information is accessed
  - Use Universal Canvas (move from HTML based presentation to XML based interaction)
  - Use Natural interface, recognizes spoken language, handwriting and understands natural language
  - Use Software services like Microsoft Passport for identity determination

What does it mean for You?

- As a Software Developer you:
    - Can use multiple languages in combination
    - Use Common Language Runtime (CRL) and Common Language Infrastructure (CLI)
    - Use .NET base classes
    - Deploy much easier (XCOPY based deployment)
    - Will debug a lot easier
    - Work with a uniform programming model both horizontal and vertical…
    - Will avoid COM, MFC, ATL, STL, RTL, Win32 API, VBScript etc.

.NET as a SW dev. Platform */basic*

- Two types of code, managed and unmanaged
- Based on CLR, Common Language Runtime
- Uniform lang. (CLS) and type system (CTS)
- Base class library abstracting Win32
- XML is key, also for database query result
- Support for building and deploying Web services
- User interface support both for Web applications and stand alone applications

.NET as a SW dev. Platform */the fun part*

- Assemblies
- Metadata
- Common Type System (CTS)
- Profiling/Debugging/Tracing support
- Memory management support
- Remoting support
- .NET as a component model
- COM / Legacy code interoperability

.NET as a SW dev. Platform */the fun part*

- Assemblies (Managed Components)
  - Basic building blocks, contain Microsoft Intermediate Language (MSIL) and Metadata
  - Self describing, no registry required
  - Unit of re-use, base for deployment, security and versioning, side-by-side usage (end of DLL Hell)
  - Can be one or more DLLs or EXEs and many other file types like resource files, GIFs etc.
  - Components are described in a manifest
  - Two types, private and shared. Shared offers global unique naming mechanism

.NET as a SW dev. Platform */the fun part*

- Metadata
  - Information in Assemblies making them self-describing. Described are: Provided types, class method and field info, version info, dependencies on other assemblies & required security attributes
  - Can be considered evolution of .TLB and .IDL
  - No need to mess around with .IDL or .H files
  - Fully available in development environment
  - Fully accessible through reflection classes and some COM interfaces
  - Together with assemblies end of DLL Hell!

.NET as a SW dev. Platform */the fun part*

- Common Type System (CTS)
  - Formal specification how .NET types look and behave
  - Types can have: Fields, Methods, Properties, Events/Delegates and Types
  - Specify visibility and member access (public, private etc.)
  - No support for multiple inheritance
  - Support for implementing multiple interfaces
  - Support for delegates
  - Support for events
  - Every type inherits from *System.Object* offering: Equals, GetHashCode, GetType, ToString, Finalize & MemberWiseClone

.NET as a SW dev. Platform */the fun part*

- Profiling support
  - Fully integrated in the CLR, *present at both development time and runtime!*
  - Profiling:
    - Create a COM class with interface *ICorProfilerCallback*
    - Interface is called for a large number of system events, function entry/exit, class load/unload, JIT compilation starts etc.
    - Together with Metadata, all information about the running program can be obtained

.NET as a SW dev. Platform */the fun part*

- Debugging/Tracing support
  - Debugging/Tracing
    - User controllable debug output and tracing with custom flags
    - Debug class with five levels
    - Trace class with five levels
    - Walking the stack is integrated in the CLR
    - Custom Context Attributes for tracing program flow, full tracing of method entry, exit, parameters etc.

```
[TraceHook]
class TracedClass
{
    // Code the rest of the class like normal!!
```

.NET as a SW dev. Platform /the fun part

- Memory Management Support
  - Memory is allocated through “new” but deleted implicitly, no delete operator present
  - Destructor doesn’t exist anymore (Finalize takes role, but..)
  - Runtime takes care of scope management
  - Objects are allocated from the managed heap
  - Memory allocations are very fast, almost as fast as stack allocations, much faster than HeapAlloc/Malloc
  - Memory is managed by Garbage Collector
  - Since a class abstracts a resource and the class instance is managed, this can also be seen as resource management
  - *Memory leaks are a thing of the past!*

.NET as a SW dev. Platform */the fun part*

- Remoting Support
  - Explicit registration with CLR required for remote objects, so no completely transparent usage
  - Communication is XML or binary based
  - Communication through channels
    - HTTP channel using SOAP protocol, binary or XML
    - TCP channel, using a binary stream
  - Marshall any possible object by value (copy)
  - After obtaining a remote object instance, using it is the same as a local object
  - Hooks offered for load balancing

.NET as a SW dev. Platform */the fun part*

- .NET as a component model
  - Advantages of COM without the pain
  - No central registration, GUIDs, IDL files, HRESULT, IUnknown, reference counting, CoCreateInstance, apartments etc.
  - No plumbing, use a class as is!
  - Object Oriented to the core, inheritance support, even across languages and process boundaries
  - Unfortunately interface based paradigm not forced!

.NET as a SW dev. Platform */the fun part*

- COM / Legacy code interoperability
  - CLR can generate managed classes to wrap COM classes
  - CLR can access regular DLLs (Win32 API)
  - “Legacy” code can access managed code through COM interfaces
  - TLBExp and TLBImp to generate assembly from type library and vice versa, RegAsm to register .NET class

Roadmap

	Today	2001	2002+
User Experience	Technology preview	Windows XP	Full .NET UE Range of devices
Infrastructure and Tools	Visual Studio.NET and .NET SDK beta	Visual Studio.NET .NET Framework	Windows.NET Server
Building Blocks	Passport & Hailstorm	3 or 4 key services	Full Offer, Corporate Federation

Some Final Thoughts

- There was a lot I did not tell you
- No revolution but an evolution
- Acceptation will happen from the bottom up
- Developing software will surely change
- Adaptation will take a few years
- Gaining knowledge about .NET is inevitable
- Doing a prototype is advisable
- Applicability in (hard) real time environment is doubtful