

# Introduction

**Tomáš Matoušek**  
&  
**Ladislav Prošek**

[tmd.havit.cz/teaching/csharp.htm](http://tmd.havit.cz/teaching/csharp.htm)

# Who Are These Guys?

- we are students of the 5<sup>th</sup> year
  - specialization: Distributed & Object Oriented Systems
- 2 years of experience with .NET Framework
  - the software project for .NET
  - conferences on .NET & VS.NET technologies
- PHP.NET software project
  - the compiler of PHP language for .NET Framework
  - continues with development
  - required detailed knowledge of FW
- far from know-alls :-0
  - your comments and feedback welcomed

# What Is the Seminar About?

- Common Language Infrastructure (CLI), .NET Framework
- C# language
- C++ with managed extensions (MC++) language
- tools on .NET
  
- theory
  - e.g. Garbage Collection algorithm, Execution Engine Internals, ...
  
- practice
  - many examples, also from PHP.NET
  
- credits for ...
  - your experience with .NET
  - simple yet interesting programs exploiting advanced features of .NET

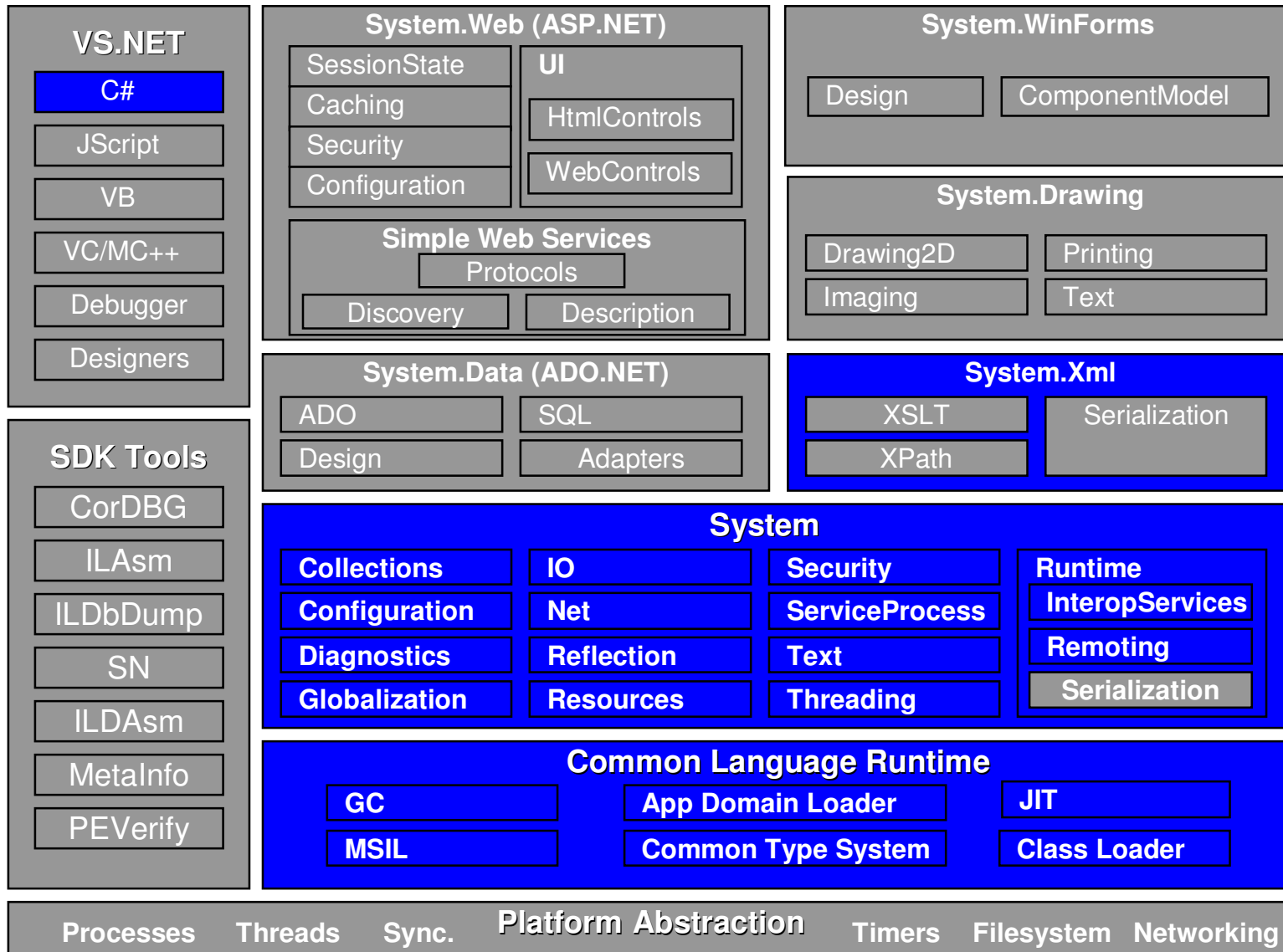
# CLI Implementations

- Microsoft .NET Framework
  - Windows 2K, XP, 2003
  - integrated with Longhorn
  - version 2.0 coming soon
- The Compact Framework
  - lightweight version of FW for PDAs, mobile phones, ...
  - Windows CE
- Mono
  - open source
  - Linux
- Rotor
  - a part of FW 1.0 released for research
  - source code available
  - platforms: Windows, FreeBSD

# Common Language Infrastructure

- Common Language Runtime (CLR)
  - execution engine: virtual machine
  - class loader, GC, verifier, ...
  - mscoree.dll
- Base Class Library (BCL)
  - classes available to all .NET languages
  - implements common functionality: strings, arrays, ....
  - mscorlib.dll
- CLI = CLR + BCL
- .NET Framework = CLI implementation + ASP.NET + ADO.NET + WinForms + Web Services + ...

# .NET Framework



# Common Language Runtime

- common runtime for many languages
  - C#, MC++, J#, Basic, Eiffel, Cobol, Fortran, Haskell, PHP, ...
- each language compiled to one universal byte code
- byte code is a machine code of VM
  - Intermediate Language (IL) – an assembler of VM
  - like JVM bytecode but more powerful
- elements of each language mapped to elements of IL
  - by a compiler of the language
  - some languages know about CLR (C#) others don't (PHP)
- each language's capability is a subset of the IL's one

# Managed vs. Native Code

- code
  - managed vs. native (unmanaged)
  - managed code – IL instructions
  - native code – CPU instructions
  - interoperability (one can call the other)
  - managed code jitted to native at runtime
- Common Intermediate Language (CIL, MSIL, IL)
  - assembler for managed code
  - stack based
  - high level instructions (e.g. newobj, ldstr, throw, callvirt, ...)
  - literals: strings, integers, reals, booleans, null
  - local variables, arguments, fields, methods, functions

# Managed vs. Native Data

- managed data (managed heap)
  - managed by Garbage Collector
  - no explicit destruction
  - freed when not reachable by managed code
  - finalization
  - very fast allocation, time consuming collections
  - sophisticated optimizations
- native data (native heap)
  - unknown for GC
  - explicit release and destruction
  - programmer should care about
  - allocations slower than managed ones
    - first fit

# Evaluation Stack vs. Native Stack

- evaluation stack
  - a stack of the virtual machine
  - each slot can contain item of arbitrary size
    - like Turing machine tape cell
    - a type of data stored in the slot is always known
  - IL instructions add/remove top items of the stack
- native stack
  - real stack, hidden to IL instructions
  - stack frames
    - arguments of methods
    - local variables
    - reflection information
  - code can reflect the current stack and see the callers
  - optimization by JITter (inlining)

# Metadata

- data describing managed data and code
- associated with all language elements
- stored together with code and data in one file
  
- compilers understand metadata
  - no need for C-like headers in managed languages
  
- programmer can
  - define and use his/her custom metadata
  - query on metadata by reflection API
  
- tools understand metadata as well
  - ILDASM, Visual Studio .NET, ...

# Assemblies

- unit of deployment (unlike Java)
- code, metadata and resources stored in assemblies
- assembly
  - one or more modules (files)
  - manifest – describing assembly content
  - executable (.exe) or library (.dll)
- private
  - used by one application only
- shared
  - used by more applications
  - placed in GAC (Global Assembly Cache)
  - strong name
    - assembly name + version + culture + public key token
    - no path
- assembly references resolved at run-time