Microsoft®
.NET

Thomas Kofler

Oktober 2012

Übersicht

- Motivation für .NET
- Was ist .NET
- Merkmale
- Ausführungsmodell
- Konzepte und Konzepte im Vergleich mit Java
- CIL, CLR, CTS, CLS
- Typsystem
- Assemblys und Metadaten
- .NET Framework 4.0
- Beispiele

Motivation für .NET

- Unzulänglichkeiten des Component Object Models
  (DLL-Hell: DLLs werden im Windows-Verzeichnis abgelegt. Nur beschränkte Möglichkeit versch. Versionen zu verwalten. Einstiegspunkt für das Programm kann sich verändert haben.)

- Scheitern der MS-Java-Strategie (J++)
  (Remote Method Invocations und Java Native Interface fehlten, dafür aber callbacks u. delegates für Event-Behandlung vorhanden – d.h. eine andere Sprache, basierend auf den Konzepten von Java.)

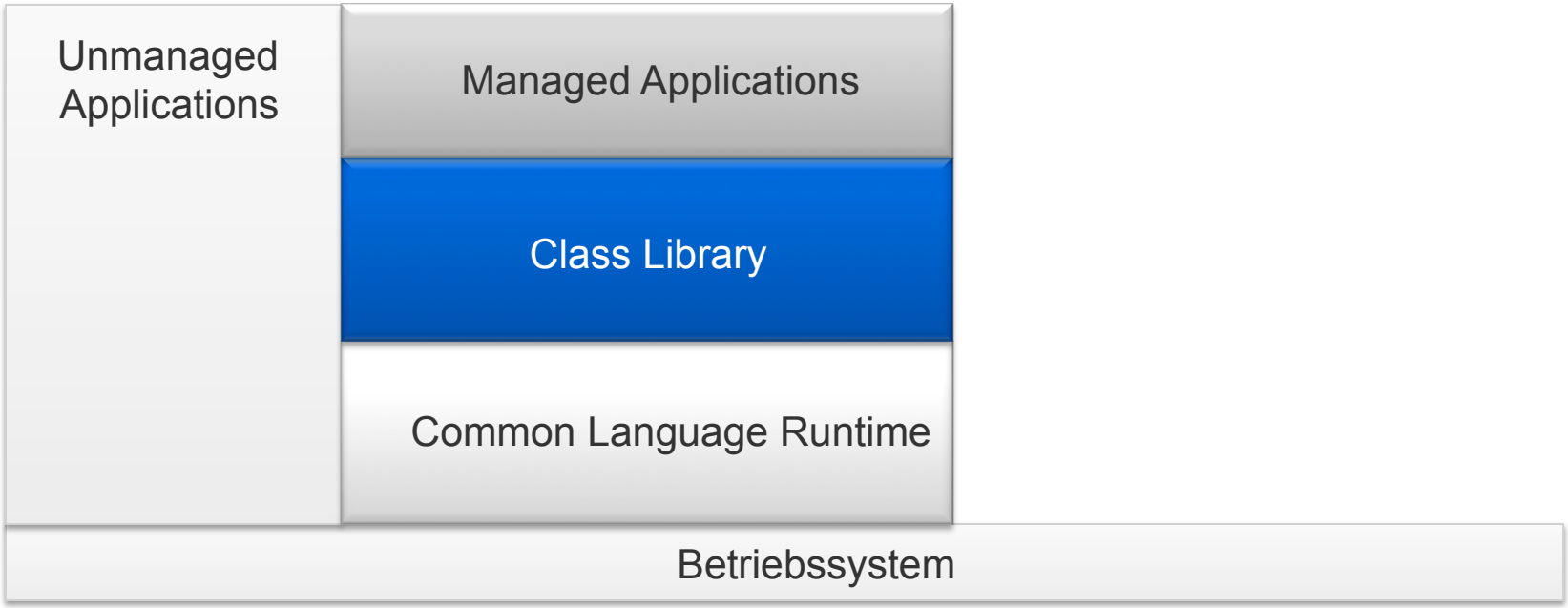
- Bedarf an moderner Programmiersprache

Motivation - Designziele .NET

- Steigerung der Entwicklerproduktivität
- Senkung von LOCs
- Einfaches Interagieren von Assemblys
  (Wiederverwendung, Kommunikation zw. Anwendungen,…)
- **Reduktion der Notwendigkeit für versch. Projekte unterschiedliche Entwicklerteams zu benötigen**
  (z.B. Web-Anwendungen, Windows-Forms-Anwendungen,….)
- **Deployment vereinfachen**
- Software-Lebenszyklus
  (z.B. Update von ClickOnce-Deployment-Anwendungen – Updates auf z.B. Netzwerkspeicher, WebServer, usw. -> autom. Update)

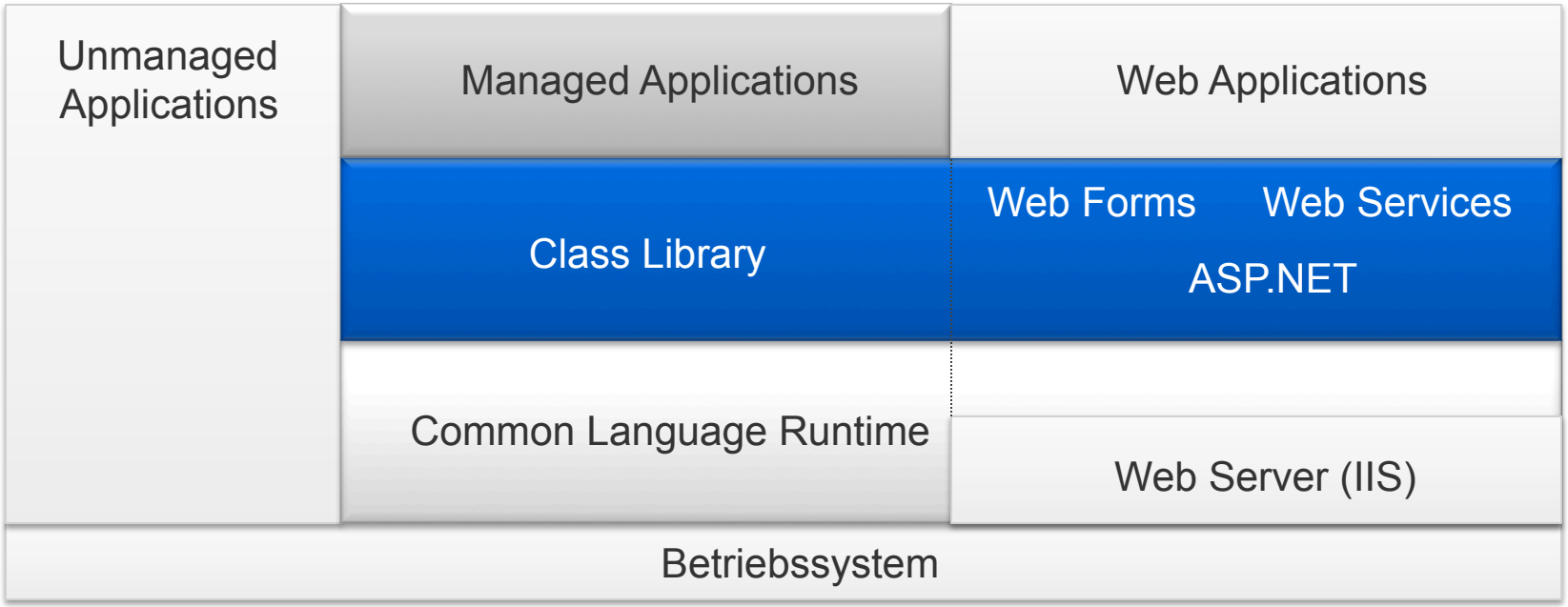
Was ist .NET?
Die Welt vor .NET war leer…

Was ist .NET?



Common Language Runtime	interoperability, security, garbage collection, versioning, ...
Class Library	GUI, collections, threads, networking, reflection, XML, ...

Was ist .NET?



ASP.NET, Web Forms	Web GUI (object-oriented, event-based, browser-independent)
Web Services	distributed services over RPC (SOAP, HTTP)

Unterstützte Plattformen

- PC/Server/Cloud

.NET Framework 4.5

- Microcontroller (Embedded Systems)

.NET Micro Framework

Unterstützte Plattformen

- PDAs/ 
  Windows Mobile

.NET Compact Framework
Teilmenge des .NET Framework

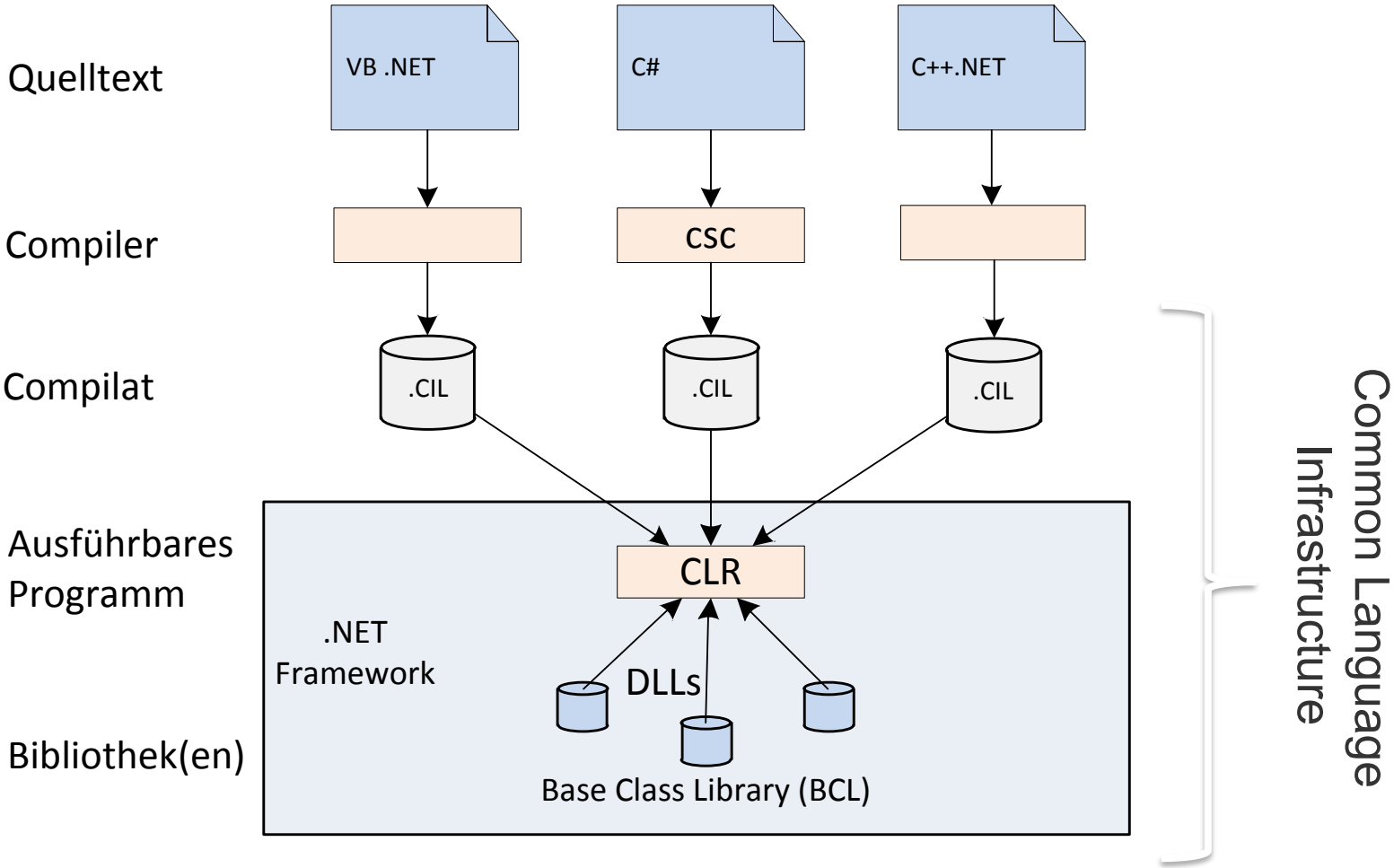
- Windows Phone 7(.5) / 8
- Windows RT Tablets

Silverlight
Teilmenge des .NET Framework bzw.
WinRT -> andere Rolle des .NET Frameworks

Merkmale

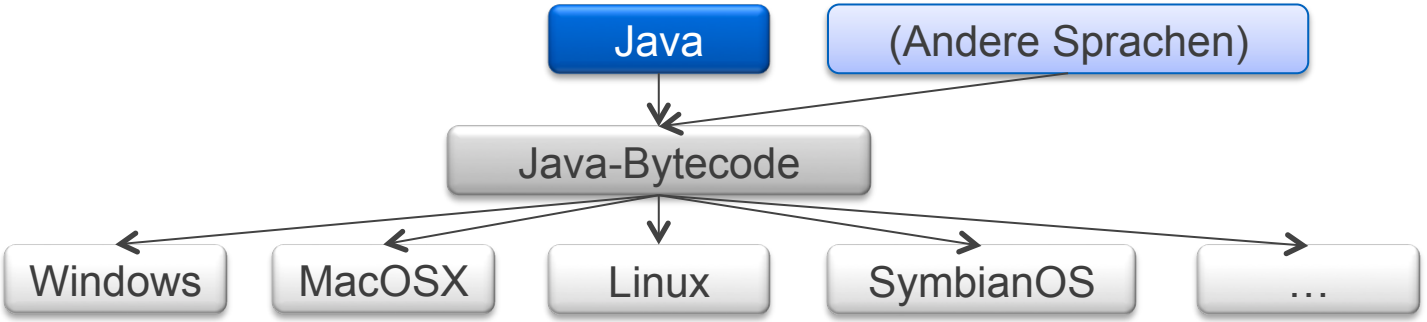
- Durchgängige Objektorientierung
- Einfach wiederverwendbare Softwarekomponenten
- Verschiedene Anwendungs-Typen
  (Konsolen-, WindowsForms- und Web-Anwendungen, Web-Services, Mobile-Anwendungen, Bibliotheken,…)
- Sprachunabhängigkeit (später mehr dazu)
- Einheitliche Laufzeitumgebung
- Umfangreiche Klassenbibliothek
  („ersetzt“ z.B. direkten Zugriff auf Windows-APIs, z.B. MFC – Microsoft Foundation Classes)
- XML-basierte Konfiguration von Anwendungen
- XCOPY-Deployment
- Interoperabilität (zu COM)

Ausführungsmodell

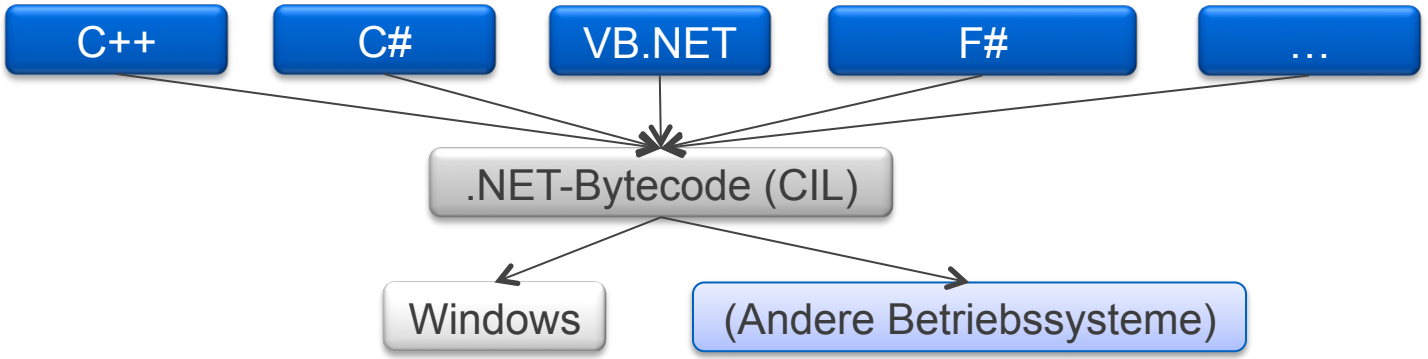


Vergleich Java mit .NET

- **Java**: Eine Sprache für viele Plattformen (cross-plattform development)



- **.NET**: Viele Sprachen für „eine“ Plattform (cross-language development)



Konzepte

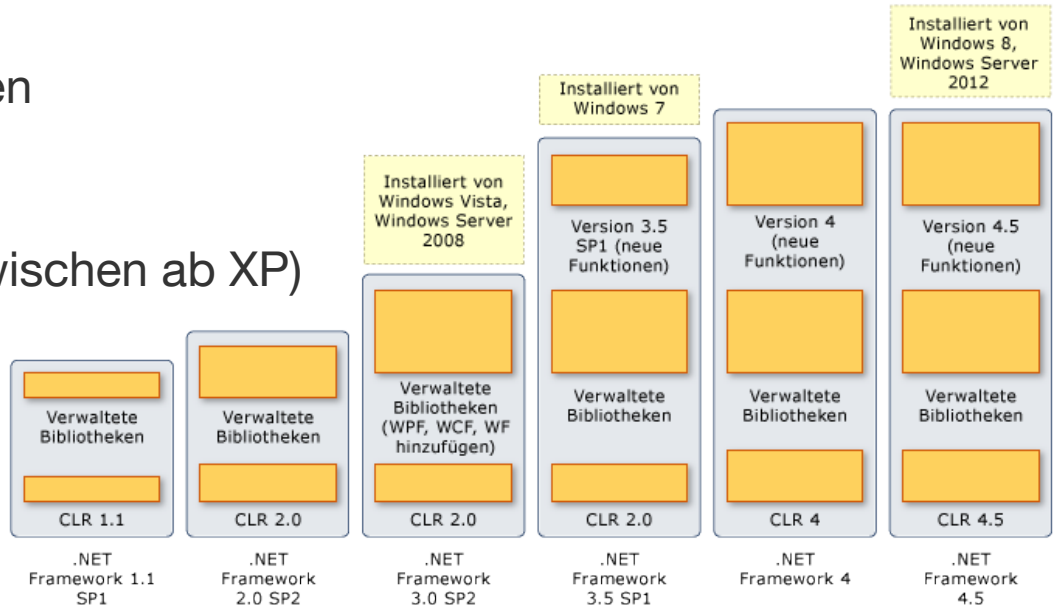
- Neues Programmierparadigma
  - Komponentenorientierte Programmierung (auf Basis der OOP)
    - Cross-Language-Calls, Cross-Language-Inheritence
    - Einheitliche Erstellung/Nutzung von Komponenten
    - Softwarekomponente == Assembly == DLL/EXE

- Entwicklungsumgebungen
  - Microsoft .NET Framework SDK
  - Microsoft Visual Studio 2012 (Express, Professional, Premium, Ultimate)

Konzepte

- Wie in Java
  - Standardbibliothek mit sehr großem Funktionsumfang (siehe Statistik später)
  - Gemeinsame Laufzeitumgebung (vergleichbar mit einer virtuellen Maschine)

- Anders als in Java
  - Viele Programmiersprachen
  - Ursprünglich nur eine Betriebssystem-Plattform (Windows, ab NT 4.0, inzwischen ab XP)
  - Mittlerweise durch Mono (www.mono-project.com) auch unter Linux und MacOSX verfügbar



CIL und CLR

- Common Intermediate Language (CIL)
  - Alle Compiler für alle .NET-Sprachen erzeugen anstelle von Maschinencode für reale Maschinen eine spezielle Zwischensprache
  - CIL-Code umfasst Programmcode und Typinformation (Metadaten)
  - CIL-Code wird nie interpretiert, sondern vor der Ausführung in Maschinencode übersetzt Just-In-Time (JIT)

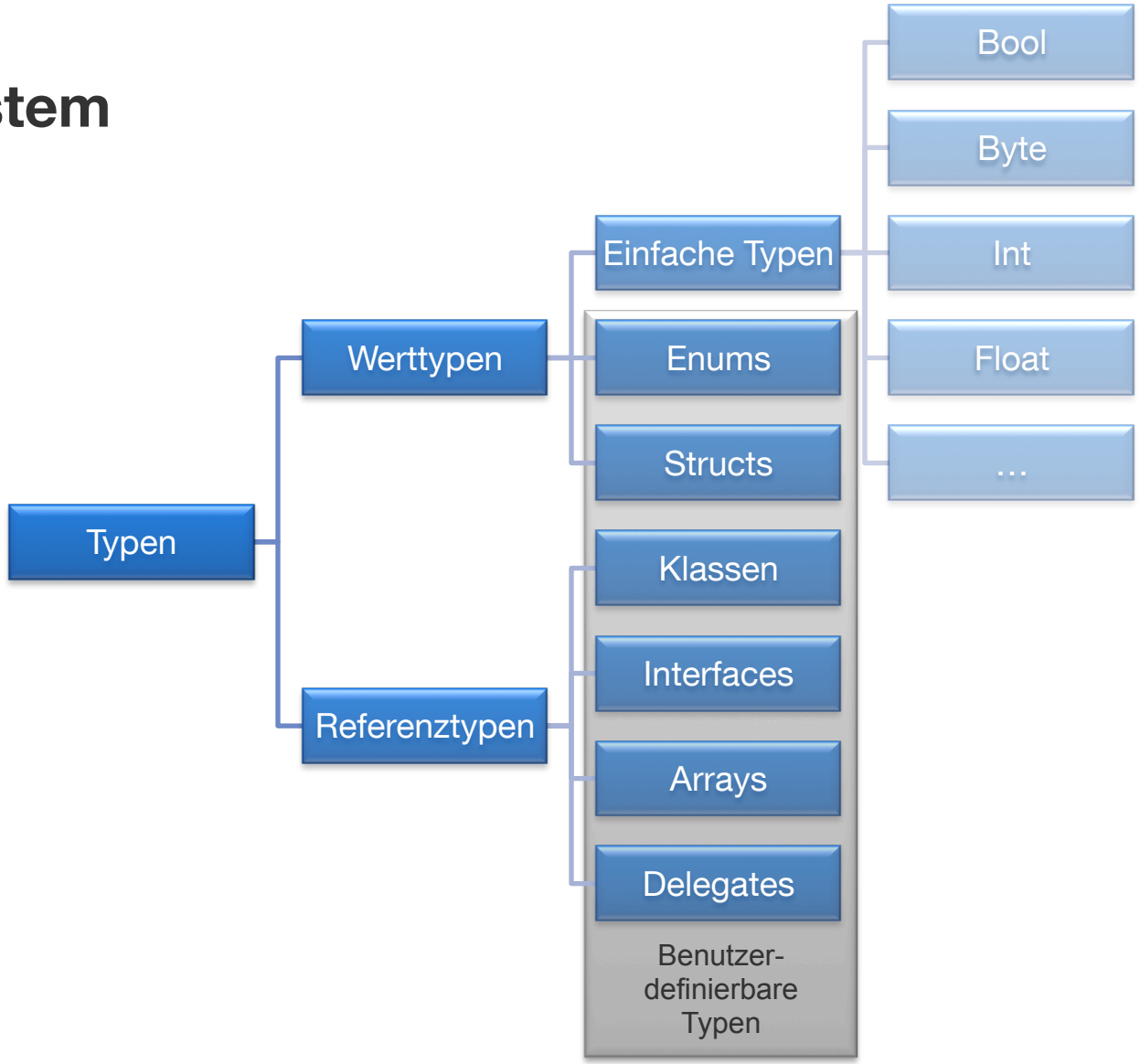
- Common Language Runtime (CLR)
  - CIL wird von einer „virtuellen Maschine ausgeführt“:
    mit automatischer Speicherbereinigung (Garbage Collection, GC),
    Ausnahmebehandlung (Exception Handling, EH), Sicherheitsmeachanismen (Signatur), Versionierung (Global Assembly Cache, GAC)
    und Interoperabilität (COM)
  - CLR besteht im wesentlichen aus drei Komponenten
    → Klassenlader (Class Loader), Code-Verifizierer (Verifier) und JIT-Compiler

CTS und CLS

- Common Type System (CTS)
  - Gemeinsames Typsystem für alle Sprachen (ermöglicht z.B. Vererbung über verschiedene Programmiersprachen hinweg)

- Common Language Specification (CLS)
  - Minimale Teilmenge des CTS, die von allen Programmiersprachen unterstützt werden muss
    - Z.B.:
      - keine vorzeichenlosen Datentypen (uint, ulong) in öffentlichen Schnittstellen
      - Keine Klassen, Methoden, Felder, usw., die sich nur durch untersch. Großschreibung unterscheiden
  - Vereinbarung zw. Sprach- und Bibliothek-Designer nur einen Teil der Sprachfunktionen öffentl. zu nutzen, die alle Sprachen gemeinsam haben

Typsystem

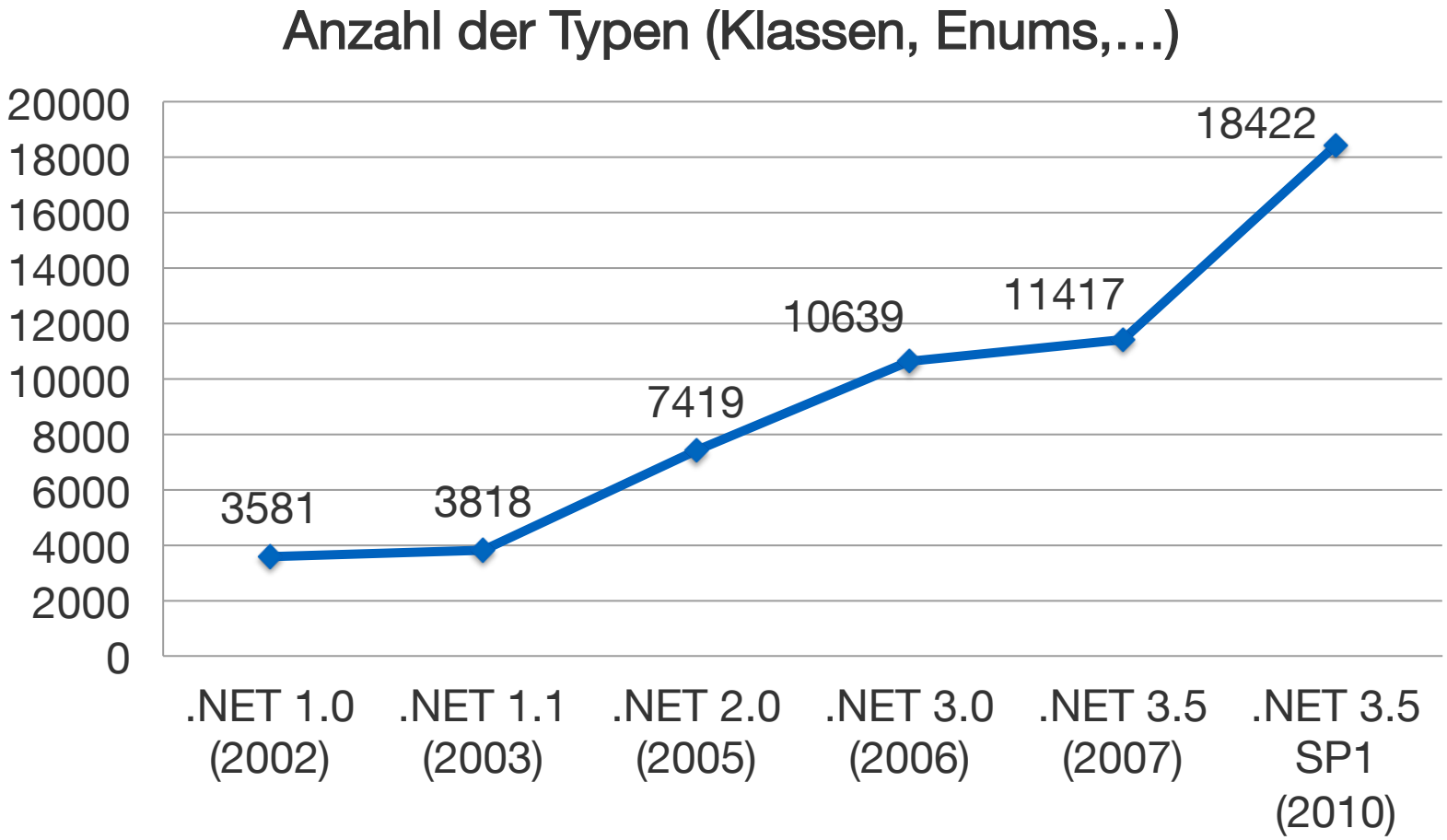

Typen
Werttypen
Referenztypen
Einfache Typen
Enums
Structs
Klassen
Interfaces
Arrays
Delegates
Benutzer-definierbare Typen
Bool
Byte
Int
Float
…

Assembly und Metadaten

- Assembly
  - .NET-Anwendung besteht aus Assemblies
    (einzeln installierbare Einheiten, in Form von DLL/EXE-Dateien)
  - Assemblies erlauben einfache Installation (Deployment)
  - Assemblies bestehen aus Manifest und ein oder mehreren Modulen (CIL-Code und Metadaten dafür)
  - CLR führt Sicherheits- und Versionsprüfung auf Basis von Assemblies durch

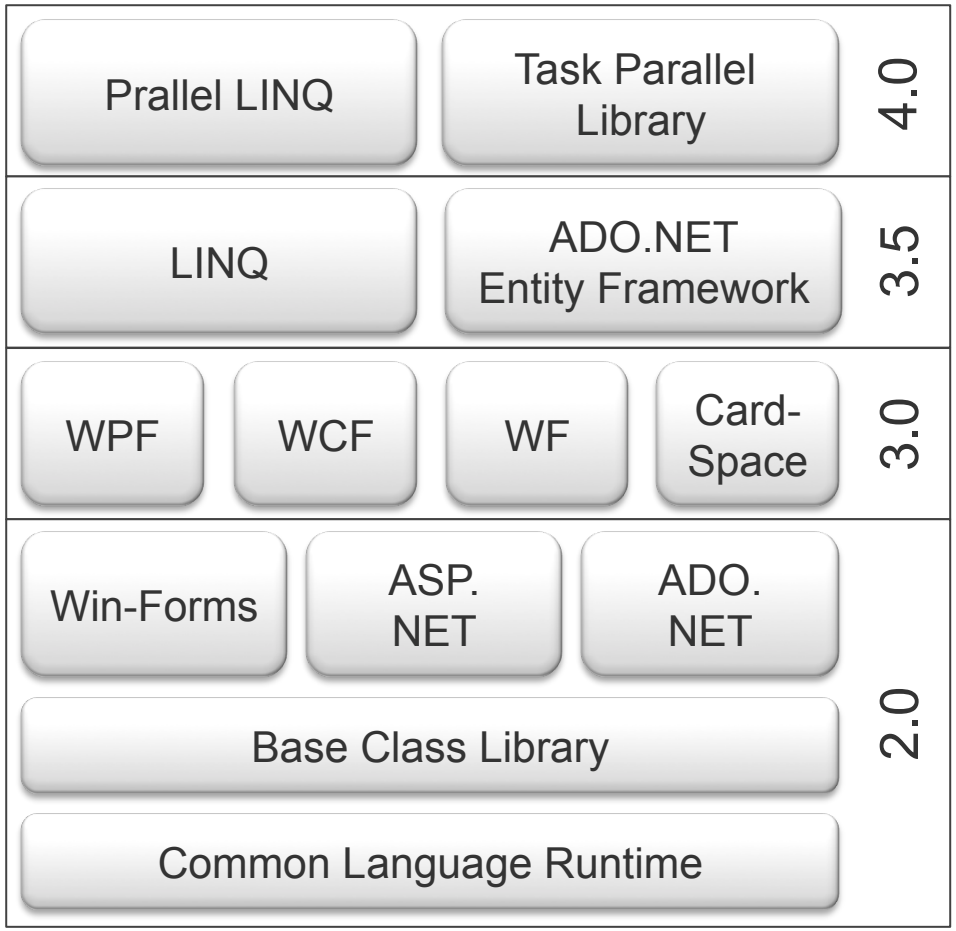
- Metadaten
  - Information über die Datentypen in einem Assembly
  - Fixer und sehr umfangreicher Bestandteil der CIL
  - Verwendung für Übersetzung, Speicher- und Sicherheitsmanagement, in diversen Werkzeugen und Anwendungen
  - Manipulation und Erzeugung von Metadaten möglich
  - Zugriff über Reflection

.NET-Framework-Versionsgeschichte (mit Vorsicht zu genießen)



Quellen: http://blogs.msdn.com/b/brada/archive/2008/03/17/number-of-types-in-the-net-framework.aspx und http://stackoverflow.com/questions/2137444/exactly-how-large-is-the-net-3-5-framework-class-library

.NET Framework 4.0



Wichtige Bestandteile des .NET Frameworks.

Von Version zu Version kommen Erweiterungen hinzu.

What's new in .NET Framework 4.5?

Windows Presentation Foundation

- Built-in Ribbon controls ★
- Databinding improvements
- Ability to add breakpoints to databindings
- Data source change aware views (Live Shaping) ★
- Validation improvements (A)
- Improved legacy UI integration
- Dispatcher improvements (A)
- Speed-up of large data sets (P)

Windows 8 support ★

- Support for Windows Runtime (WinRT)
- .NET Profile for Metro-style apps
- Improved support for sharing DLLs between .NET profiles

ASP.NET

- ASP.NET Web API - a new framework for REST endpoints ★
- Support for implementing WebSocket receivers ★
- Built-in JavaScript + CSS combining and minification ★

ASP.NET MVC 4

- Async controllers (A)
- Built-in mobile templates + jQuery.Mobile support
- Alternate views (e.g. print version, mobile site)
- Support for Recipes: intelligent codegen ★

ASP.NET Web Pages 2

- New site templates
- Versatile validation support
- Support for OAuth and OpenID ★
- Built-in map embedding tools; supports Google, Bing and others

Did you know? ASP.NET Web Pages is yet another way to work the web besides Web Forms and MVC. WP sites use Razor and are typically developed with WebMatrix.

Web Forms

- Strongly typed data binding
- MVC-like support for Models ★
- HTML encoded binding expressions

HTML5 support ★

- Control support for new semantic elements
- Multifile support for FileUpload control
- Validator and UpdatePanel now support new HTML5 elements

Tooling

- Markup editors improved (a lot)
- IIS Express used by default
- Page Inspector ★
- New templates and snippets

- **Asynchronous pipeline support** (A) (Response, Request, HttpHandlers)
- **Performance improvements**: Multicore JIT, 35 % faster startup, memory optimizations, assembly sharing between sites, pre-fetch support (P)
- **Request validation improvements**: AntiXSS built-in, validation usable per field

Windows Communication Foundation

- New channels (UDP multicast, WebSockets, ...) ★
- Support for contract-first development ★
- Asynchronous operations (A)
- Streaming improvements
- Simplified configuration, VS config validation
- Multiple auth modes for HTTP endpoints
- New, simple HttpClient class

Windows Workflow Foundation

- C# Expressions
- State machine workflows are back! ★
- Workflow versioning ★
- Code-first activity design
- Faster execution (P)
- Designer usability improvements

Managed Extensibility Framework 2.0

- Support for generic parts ★
- Debugging improvements
- Support for explicit and convention-based bindings between objects ★
- Support for binding POCOs: no more attribute requirements

ADO.NET

- Sparse columns support improved (SQL Server)
- Passwords are now stored encrypted
- Asynchronous operations (A)

SQL Express LocalDB

New light version of SQL Express for developer use. Supported in .NET 4.5, separate patch for 4.0 is coming.

SQL Server 2012 ("Denali") Support

- High Availability support on connection string level
- Fast failover across multiple subnets
- Support for new spatial data types (polygons, arcs etc.) ★

Entity Framework 4.5

- Enumeration support
- Migrations for schema changes ★
- Designer improvements
- Spatial data type support ★
- Table-valued function support ★
- Multi-result sproc support
- Multiple diagrams per model
- Code-first support ★
- Auto-compiled LINQ queries (P)

Base Class Library

- Networking improvements (IPv6 enhanced, IDN, EAI etc.)
- Key interfaces (e.g. file IO) now support async (A)
- New ArraySegment and ReadOnlyDictionary classes
- Support for CLR objects over 2 GB in size
- Resource file management performance improved (P)
- Unicode improvements (v6, console support)
- Background JIT on multicore platforms (P)

Task Parallel Library (A)

Task thread controls improved: Task.WaitAll/.WaitAny, various timeout primitives available

TPL Dataflow: Tools for parallel data flow processing ★

C# 5.0

- Support for async programming: async and await keywords (A)
- Methods can access call site info as parameters (CallerInfo)

Visual Basic 11

- Iterator implementations (Yield)
- Async and Await equal to C# (A)
- "Global" keyword for namespace referencing
- Call Hierarchy view available
- CallerInfo attributes (as in C#)

F# 3.0

- Type providers ★
- Query expressions (LINQ) ★
- Auto-implemented properties

Visual C++ 11

- C++11 standard support improved
- Auto-vectorization and parallelization of loops (A) (P)
- GPU-driven processing (C++ AMP)
- Intellisense for C++/CLI

Legend:

- (A) = Asynchrony support
- (P) = Performance improvement
- ★ = Significant new feature

Microsoft tech·days Helsinki 2012

SANKO
Suomen Aktiivisten .NET-kehittäjien Kerho
www.facebook.com/sanko.net

offbeat solutions
www.offbeat.fi

Information based on public sources available updated on 1st March 2012 (.NET 4.5 beta).

Composed by Jouni Heikniemi.
Original with updates available @ www.heikniemi.net/hardcoded

http://www.heikniemi.net/hardcoded/2011/10/whats-new-in-net-framework-4-5-poster/

.NET Framework nach Anwendungsgebiet

- UserInterface
  - ASP.NET
  - WPF
  - WinForms
- Services
  - Data Services (OpenDataProtocol -> Abfragen/ Veränderungen über HTTP)
  - WCF
  - „Velocity“ (Skalierbarer Anwendungscache -> verteiltes Caching f. große ASP.NET Anw.)
  - Windows Workflow Foundation
- DataAccess
  - ADO.NET, Entity Framework
  - LINQ to SQL
- Core
  - Parallel Extensions (Parallel LINQ, Task Parallel Library)
  - Managed Extensibility Framework (Infrastruktur für die Entwicklung erweiterbarer Anwendungen)
  - LINQ
  - Dynamic Language Runtime (Unterstützung dynamischer Sprachen)
  - Base Class Library

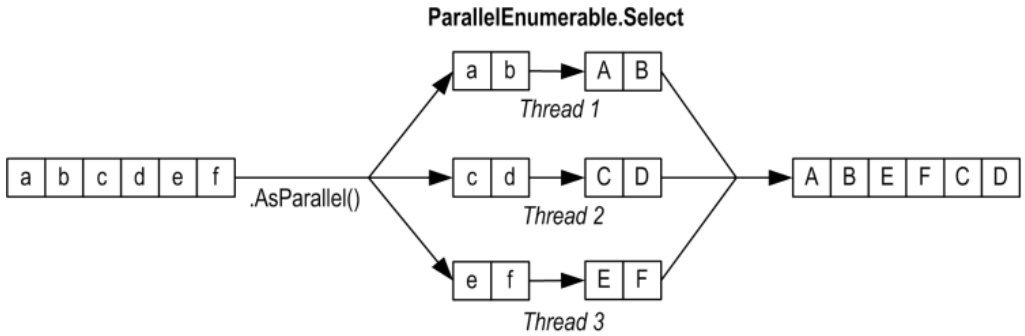
Beispiele Verwendung .NET Frameworks

- Parallel-Extension (Erweiterung der Bibliothek)
  Abstraktion der Parallelisierung
  - Code wird weiter ausgeführt, sobald die Kalkulation beendet ist

Task Parallel Library

```
1 int calculations = 100;
2 ComplexCalculationResult[] results
3             = new ComplexCalculationResult[calculations];
4 Parallel.For(0, calculations, i => {
5     results[i] = ComplexCalculation(i);
6 });
```

Parallel LINQ




```
"abcdef".AsParallel().Select (c => char.ToUpper(c)).ToArray()
```

(Quelle: http://www.albahari.com/threading/part5.aspx = Sehr gute Quelle für Infos über die Parallel-Extension)

Beispiele Verwendung .NET Frameworks

- LINQ (Evolution der Programmiersprache C#, Lambda-Ausdrücke[1])
  Language Integrated Query
  - Es wird formuliert, **was man will** und nicht, wie

```
1 var v = (from c in Center.ControlableComponents
2         where c.IsActive && c.HasNextFile
3         orderby (c.Turns * c.Priority) ascending
4         select c).FirstOrDefault();
5 return v;
```

In diesem Beispiel werden von einer Collection (Center.ControlableComponents) alle jene Elemente gewählt, die aktiv sind (c.IsActive) und eine nächste Datei in der Warteschlange haben (c.HasNextFile). Sortiert wird nach wie oft die Komponente bereits dran war (c.Turns) multipliziert mit der Priorität (c.Priority). Aufsteigend sortiert.

Die Erweiterungsmethode „FirstOrDefault“ gibt das erste Element der Rückgabeliste zurück oder das Default Elemente (bei einem Referenztypen ist das „null“).

[1] http://msdn.microsoft.com/de-de/library/bb397687.aspx

Ende