.NET als alternative Middleware-Technologie

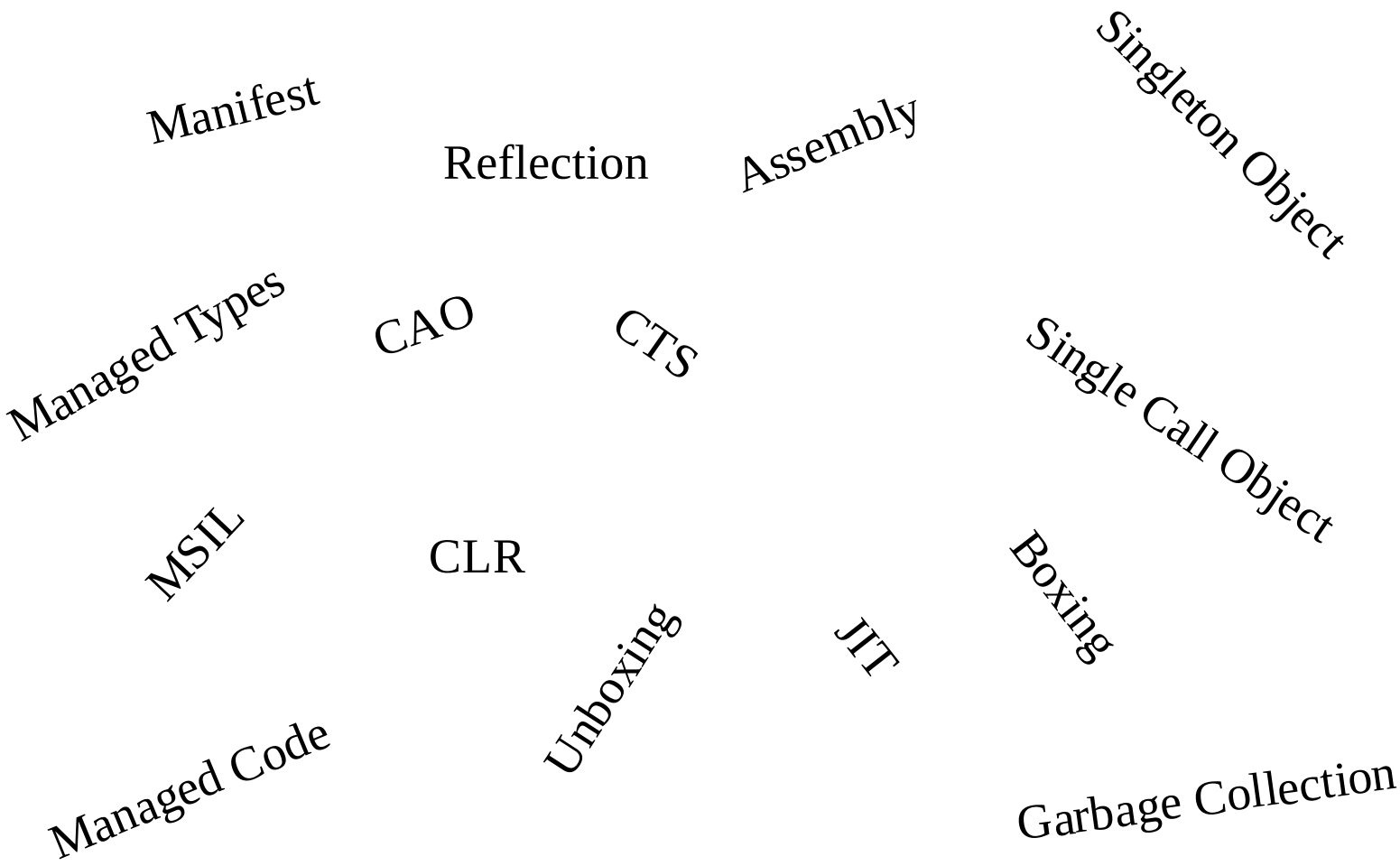
Einführung und Überblick

Vortragender: Jens Zimmermann
Veranstaltung: Hauptseminar Telematik

Inhalt

- Motivation
- Einführung in .NET
- Historische Entwicklung
- Bestandteile der .NET-Plattform
- Das .NET-Framework
  - Die Common Language Runtime
  - Die gemeinsame Klassenbibliothek
- Web Services
- .NET-Remoting
  - Technische Grundlagen
  - Beispiel
- .NET im Vergleich mit CORBA
  - Beispiel
- Literatur

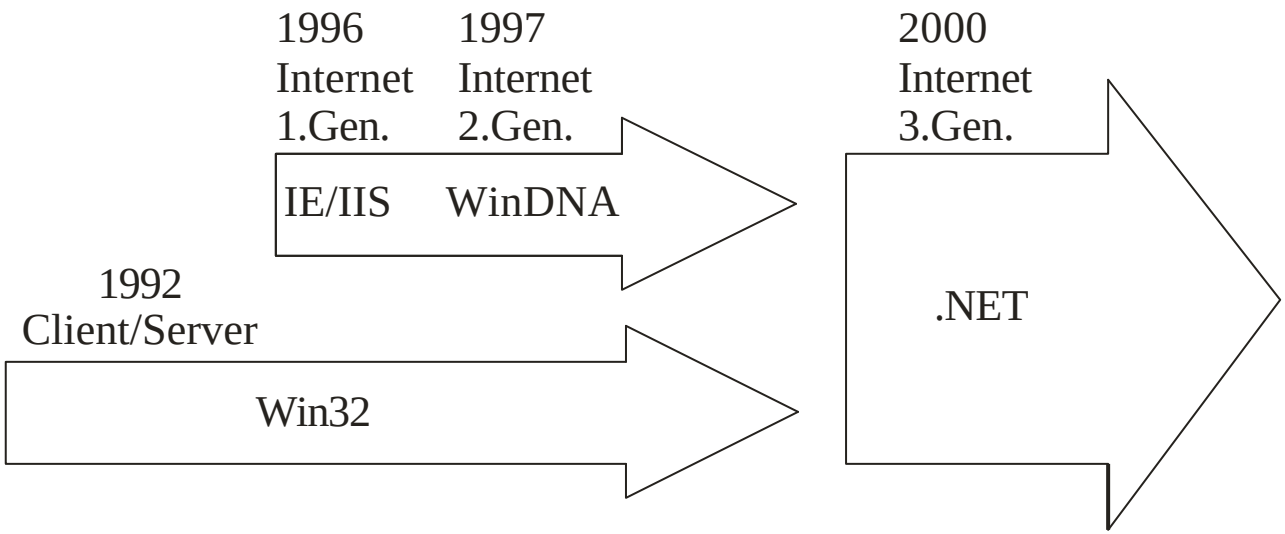
Motivation


Manifest
Reflection
Assembly
Singleton Object
Managed Types
CAO
CTS
Single Call Object
MSIL
CLR
Boxing
Unboxing
JIT
Managed Code
Garbage Collection

Einführung in .NET

- .NET laut Microsoft:
  „Ein neues Architekturmodell und eine Plattform für die nächste Generation der internetzentrierten Informationsverarbeitung“
- .NET-Strategie:
  - Verbindung PC – Mobiltechnologie (Handy, PDA etc.)
  - Tragende Rolle als Kommunikationsmittel: Das Internet
- .NET besteht aus umfangreichen Satz von Werkzeugen zur Softwareentwicklung:
  - fürs Internet,
  - für Windows-PCs und
  - für alle erdenklichen anderen Gerätschaften
- Entwicklung von .NET-Anwendungen:
  - .NET Framework SDK
  - Visual Studio .NET

Historische Entwicklung



1. Generation 1994-1996	2. Generation 1996-2000	3. Generation 2000+
Statische Seiten	Dynamische Seiten	Nutzung von Diensten
E-Mail, Basisinformationen	Personalisierung, E-Commerce	Web Services
IE/IIS	Windows DNA	Microsoft .NET

Bestandteile der .NET-Plattform

- vier wesentliche Bestandteile der .NET-Plattform:
  - Framework und Tools
  - Building Block Services
  - Enterprise Server
  - Mobile Devices
- Bestandteile des .NET-Frameworks:
  - Common Language Runtime (CLR)
  - die gemeinsame Klassenbibliothek
  - Active Server Pages (ASP.NET)
  - (Entwicklungsumgebungen)
  - (Web Services)
  - (.NET Remoting)

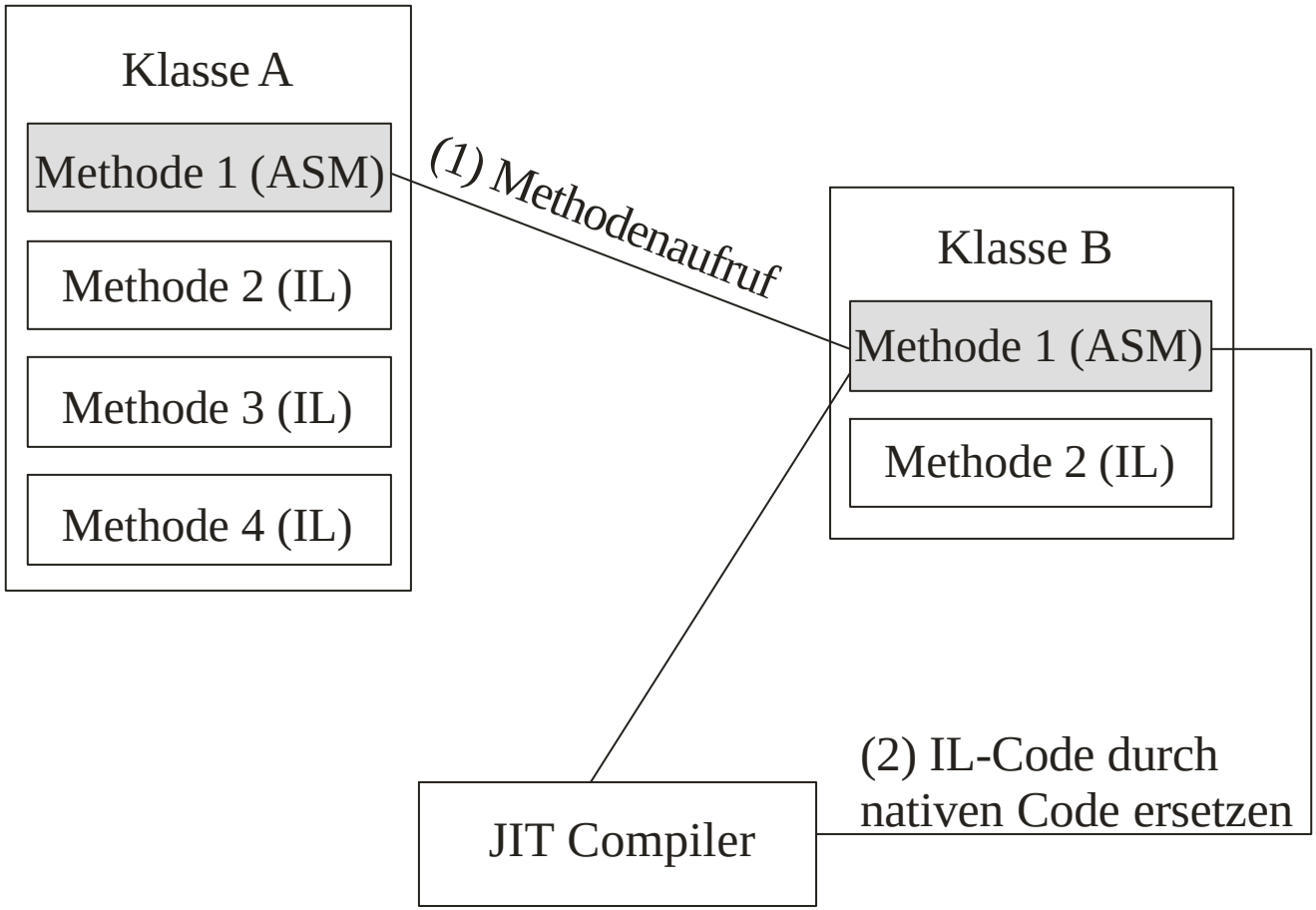
Common Language Runtime 1/6

- Nachteil der herkömmlichen Modelle zur Kommunikation und Integration von Softwarekomponenten:
  - Integration eines Layers zur Implementierung des Typsystems
  - Anpassung der Sprache an das Modell
  - Konvertierung von Sprachtypen
  - Implementierung von Aufrufkonventionen
  - Code dadurch komplex und fehleranfälliger
- durch CLR einheitliches Integrationsmodell

Common Language Runtime 2/6

- Compiler erzeugen plattformunabhängigen Zwischencode
- Zwischensprache: Microsoft Intermediate Language (MSIL)
- nur Visual C++ kann nativen Code erzeugen
- Ausführung durch Laufzeitumgebung (CLR)
- Code der unter Aufsicht der CLR ausgeführt wird: Managed Code
- bei Ausführung Umwandlung des Zwischencodes in nativen Code durch Just-in-time-Compiler
- Delegierung von bestimmten Aktionen an CLR:
  - Anlegen eines Objekts
  - Aufruf einer Methode

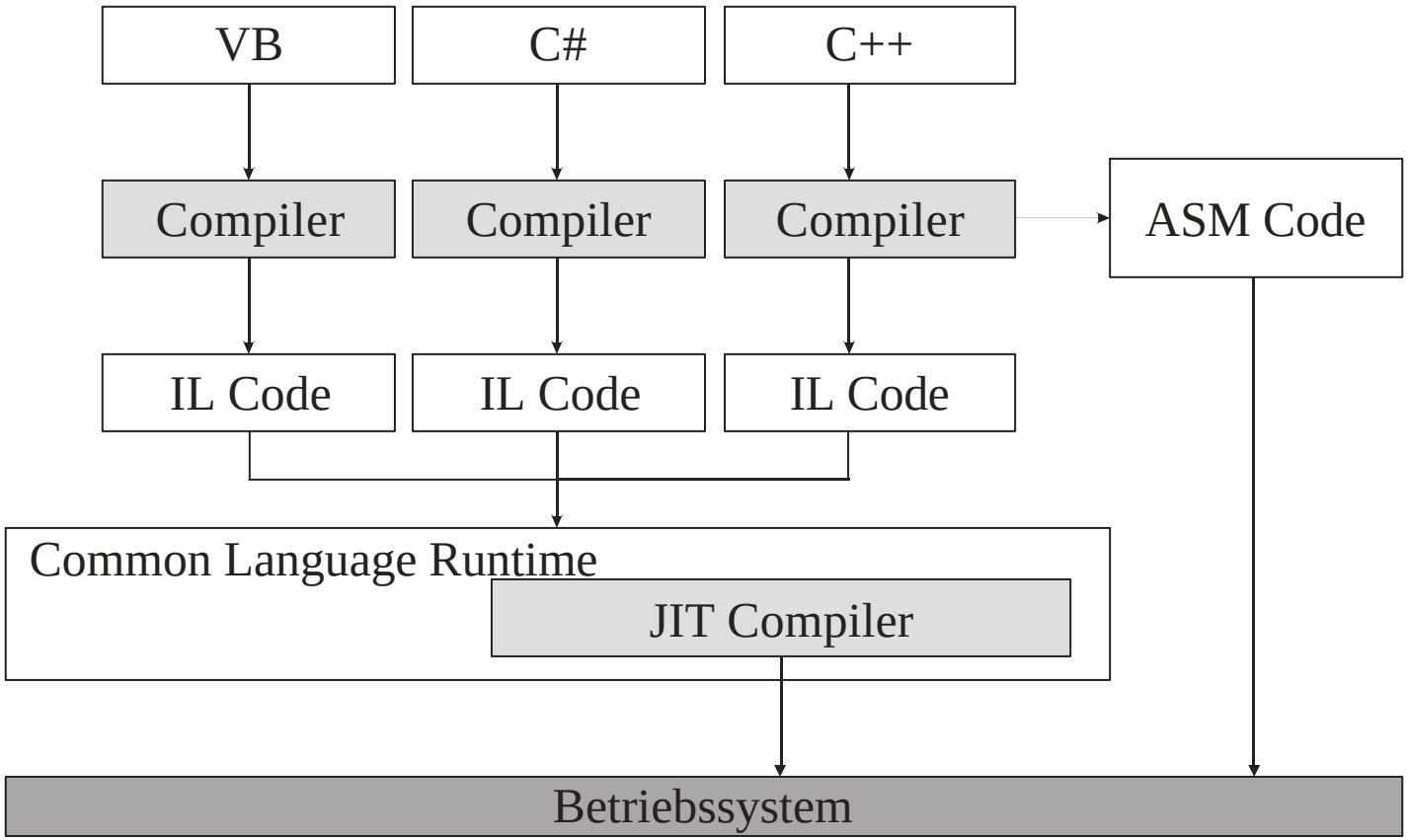
Common Language Runtime 3/6



Common Language Runtime 4/6

- Framework SDK und Visual Studio .NET bieten Compiler für:
  - Visual C++
  - Visual Basic
  - C#
- Entwicklung von Compilern durch Fremdanbieter u.a. für:
  - Cobol
  - Pascal
  - Eiffel
  - SmallTalk
- Pascal-Compiler im Beta-Stadium im Internet verfügbar

Common Language Runtime 5/6



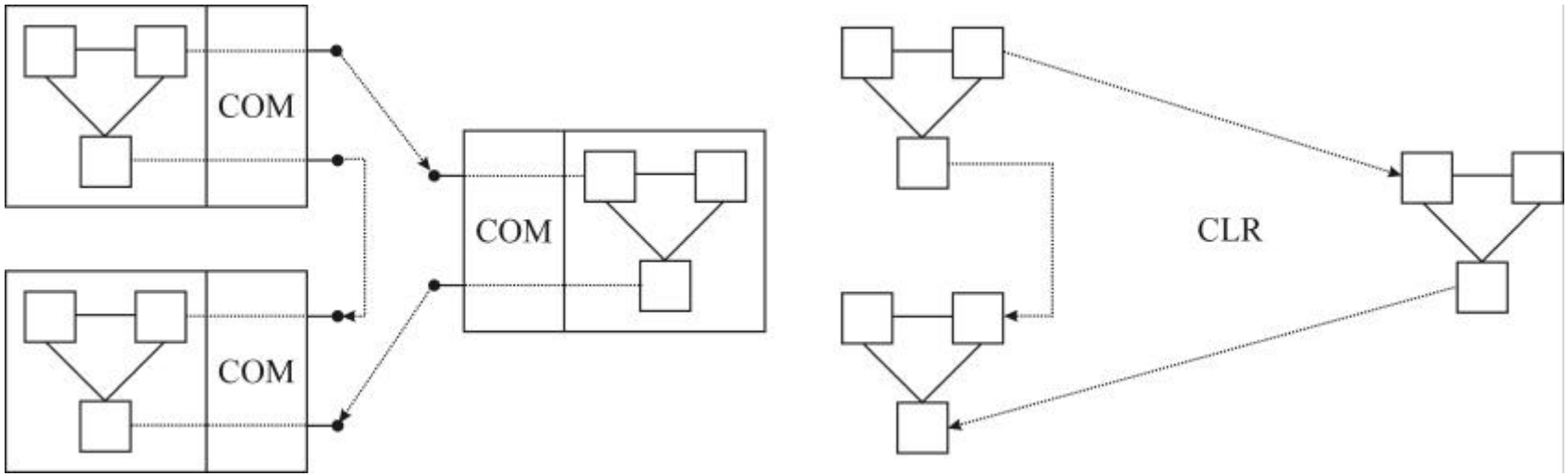
Common Language Runtime 6/6

- Vorteil der Integration auf Codeebene:
  - Programmierer kann mit der Sprache entwickeln, die ihm am besten liegt, Voraussetzung: Compiler kann MSIL-Code erzeugen
  - Ableitung und Nutzung von Klassen, die in anderen Sprachen entwickelt wurden, möglich
- Speicherverwaltung über Garbage Collection
- CLR ist vollständig offen gelegt und standardisiert
- plattformunabhängige Entwicklung durch CLR, aber Implementierung nur für Microsoft-Betriebssysteme
- CLR für Linux in Entwicklung (Projekt Mono – http://www.go-mono.org)

Common Type System 1/5

- ein Typsystem für alle Sprachen: Common Type System (CTS), wird durch CLR zur Verfügung gestellt
- Vorteile:
  - keine unterschiedlichen Repräsentationen eines Datentyps
  - einheitliche Größe eines Datentyps
  - keine Konvertierung von Zeichenketten notwendig
  - problemlose Kommunikation von verschiedenen Komponenten
- Konvertierung und Anpassung an COM-Aufrufkonventionen entfällt
- Kommunikation mit COM, COM+ und DCOM-Komponenten aber weiterhin möglich (Mapping)

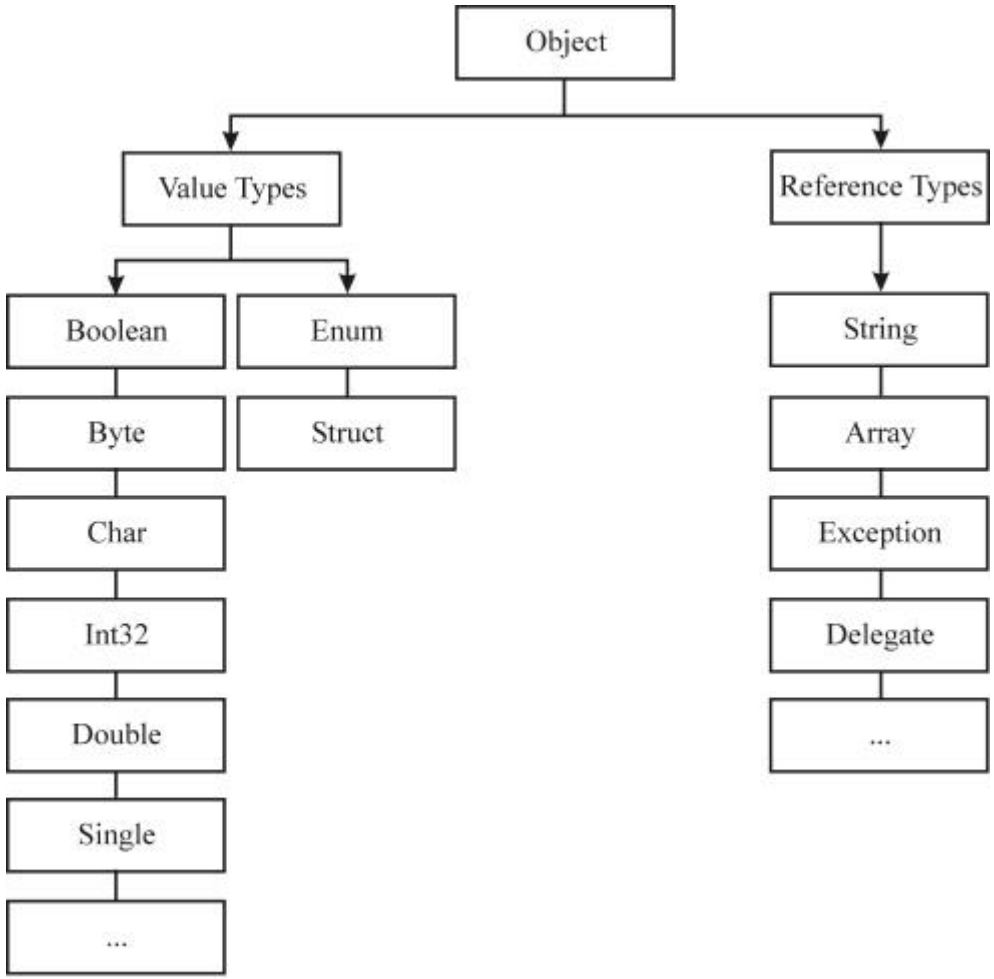
Common Type System 2/5



Common Type System 3/5

- Datentypen des CTS werden als Managed Types bezeichnet, alle von System.Object abgeleitet
- alle Typen sind Objekte, zwei Arten:
  - Value Types:
    - auf dem Stack angelegt
    - enthalten Daten
    - können nicht den Wert „Null“ annehmen
    - im wesentlichen primitive Datentypen, Aufzählungen und Strukturen
  - Reference Types:
    - auf dem Heap angelegt
    - enthalten Referenzen auf Objekte
    - können den Wert „Null“ annehmen
    - im wesentlichen Klassen, Zeichenketten und Felder

Common Type System 4/5



Common Type System 5/5

- kein Methodenaufruf bei Objekten auf dem Stack möglich
- Anlegen eines temporären Objekts durch CLR und Kopieren der Werte
- Technik wird als Boxing bzw. Unboxing bezeichnet
- durch Compiler werden bei Übersetzung Metadaten in Komponente geschrieben, welche Beschreibung sämtlicher Typen der Komponente enthalten
- Auslesen dieser Metadaten: Reflection

Assemblies 1/2

- größtes Problem der COM-Welt: Versionierung von Komponenten
- Komponenten werden von mehreren Anwendungen genutzt, Probleme bei Aktualisierung oder Veränderung: „DLL-Hölle“
- alle Komponenten, die eine .NET-Anwendung benötigt, werden als Assembly bezeichnet
- Metadaten beschreiben die Abhängigkeit der Komponenten (Manifest)

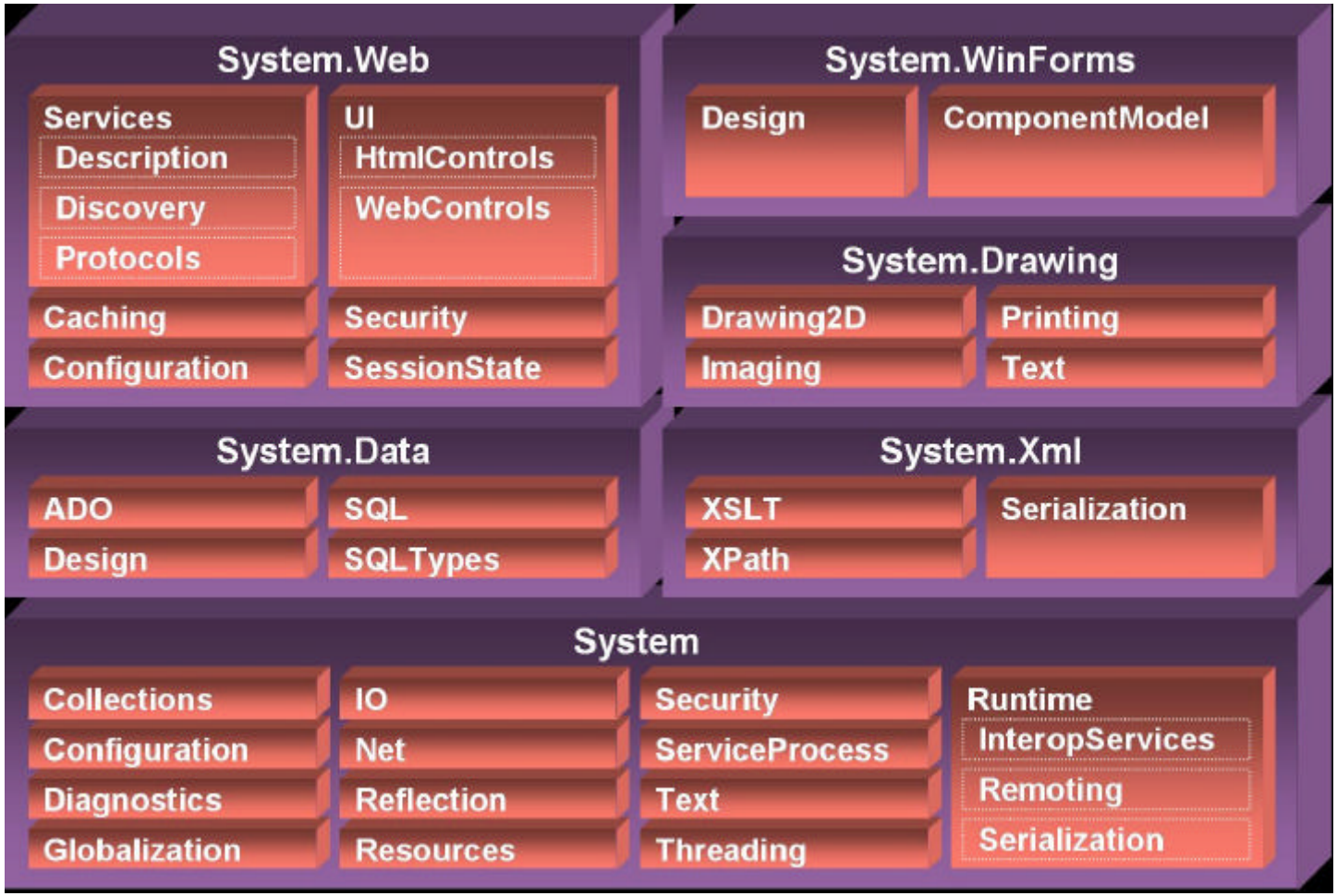
Assemblies 2/2

- Private Assembly:
  - Komponenten befinden sich im Verzeichnis der Anwendung
  - Konfigurationsdatei (.cfg) im XML-Format gibt Ort an
  - Konfigurationsdatei mit gleichem Namen und gleichem Ablage-ort, wie Anwendung
  - Nachteil: Redundanzen von Komponenten
- Shared Assembly:
  - Komponenten können von mehreren Anwendungen genutzt werden
  - Komponenteninformationen nicht mehr systemweit gültig, sondern in Konfigurationsdateien anwendungsspezifisch

Klassenbibliothek von .NET 1/2

- alle .NET-Compiler nutzen selbe Klassenbibliothek
- Zugriffe auf Betriebssystem und „Win32-API“ werden über Klassen abstrahiert
- Unterbereiche der Klassenbibliothek System:
  - WinForms (grafische Windowsanwendungen)
  - Web (Web Services)
  - Datenbankarbeit (Data)
  - XML
  - Drawing
- Klassenbibliothek ist nur teilweise offen gelegt und wird nicht standardisiert

Klassenbibliothek von .NET 2/2



Web Services

- Anwendung, die ihre Methoden über Schnittstellen nach außen zur Verfügung stellt
- Implementierung für Client vollständig transparent
- Web Service wird über URL angesprochen
- Methodenaufruf erfolgt über SOAP
- keine homogene Infrastruktur notwendig
- Nachricht wird per XML verpackt und mittels HTTP an Server geschickt, Server antwortet mit XML-Nachricht
- Kombination von Web Services zu Web-Applikationen möglich

.NET-Remoting 1/9

- ermöglicht Objekten die anwendungsdomänenübergreifende Kommunikation miteinander
- Arten von Remoting-Objekten:
  - Clientaktivierte Objekte (CAO)
  - Serveraktivierte Objekte
- unterschiedliche Lebensdauerverwaltung, bei CAO durch Lebensdauer-Manager
- Unterteilung von Serveraktivierten Objekten:
  - Single Call
  - Singleton

.NET-Remoting 2/9

- Basisklassen für Remoteobjekte:
  - MarshalbyRefObject
  - MarshalbyValue
- nach Aktivierung eines Remoteobjekts erfolgt Erstellung eines Proxy-Objekts
- Aufgaben des Proxy-Objekts:
  - Vermittler zwischen Client und Remote-Objekt
  - Weiterleitung der Aufrufe
- Nachrichtentransport über Channel-Objekte:
  - Transport von Parametern des Methodenaufrufs zum Remoteobjekt
  - Transport des Ergebnisses vom Remoteobjekt zum Client

.NET-Remoting 3/9

- Remoteobjekte nutzen Channel gemeinsam
- Registrierung der Channel durch Serveranwendung
- nach Aktivierung überwacht Channel den Port
- Speicherung der Referenz auf Remote-Objekt nach dessen Erzeugung in Tabelle
- bei Objektanforderung wird Tabelle nach Objekt durchsucht und gegebenenfalls angelegt
- Anpassung von Channels u.a. möglich für:
  - Autorisierung
  - Verschlüsselung

.NET-Remoting 4/9

- Instanzen, die eine Nachricht passiert:
  - SecuritySink
  - TransportSink
  - FormatterSink
- Arten von Channels:
  - HTTP-Channel
    - Transport über HTTP-Protokoll zum Ziel-URI
    - Konvertierung vor dem Transport nach XML durch SOAP-Formatierer
  - TCP-Channel
    - Transport über TCP-Protokoll zum Ziel-URI
    - Verwendung eines Binärformatierers

.NET-Remoting 5/9

- Registrierung aller Remoteobjekte im Remoting-Framework durch Hostinganwendung
- Informationen für Registrierung:
  - Name des Assembly
  - Typ des Remoteobjekts
  - Objekt-URI
  - Objektmodus für Aktivierung
- Registrierung über:
  - RegisterWellKnownType
  - ConfigureRemoting
- Instanziierung eines Objekts erst beim ersten Methodenaufruf

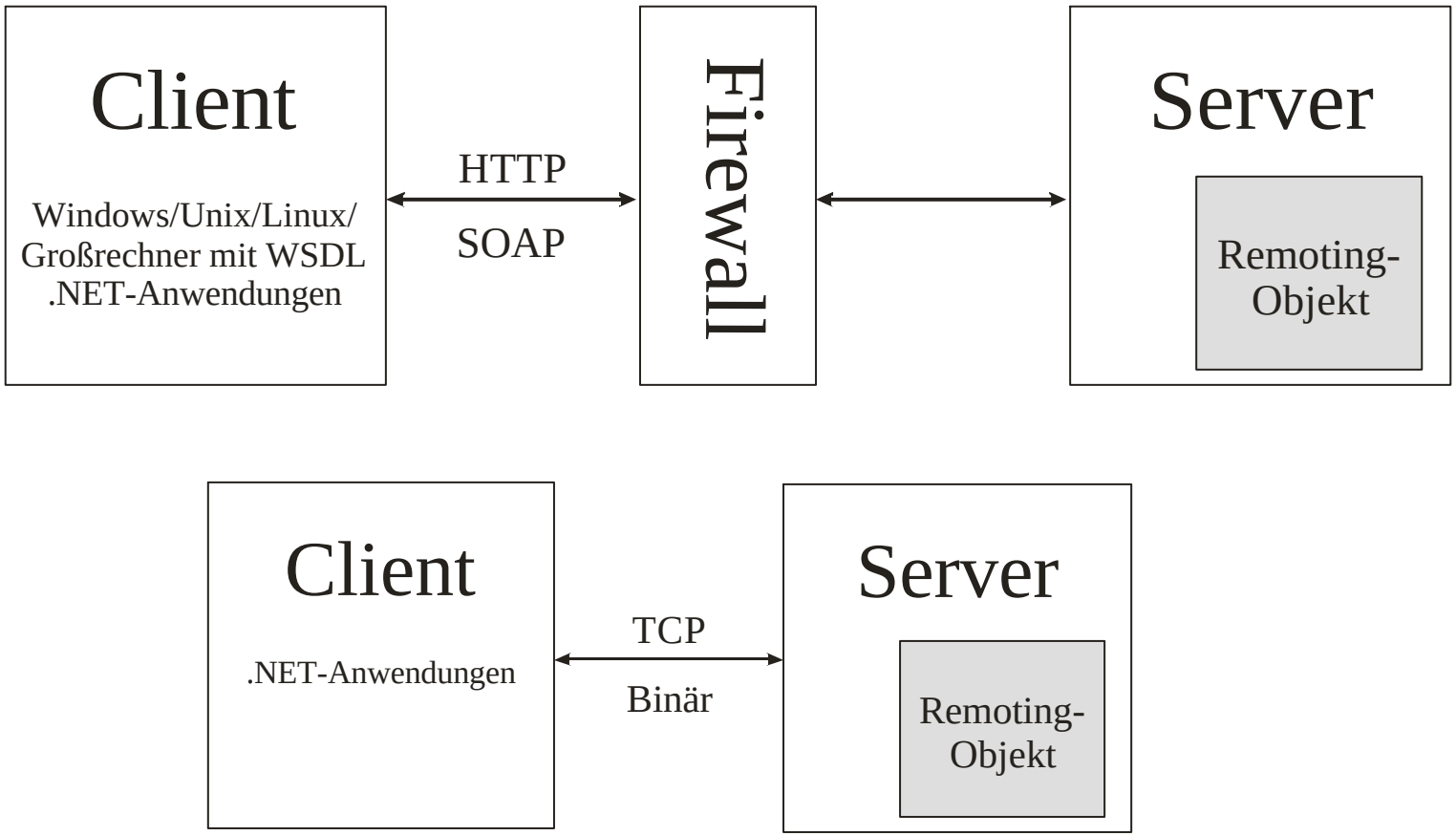
.NET-Remoting 6/9

- Bereitstellung von Typinformationen zum Remoteobjekt:
  - Verweis auf das Assembly
  - Erzeugung einer Schnittstellenklasse
  - Extrahieren der Metadaten aus dem Endpunkt durch SOAPSUDS
- Hosting von Remoteobjekten durch:
  - verwaltete ausführbare Datei
  - Internet Information Server
  - .NET-Komponentendienst

.NET-Remoting 7/9

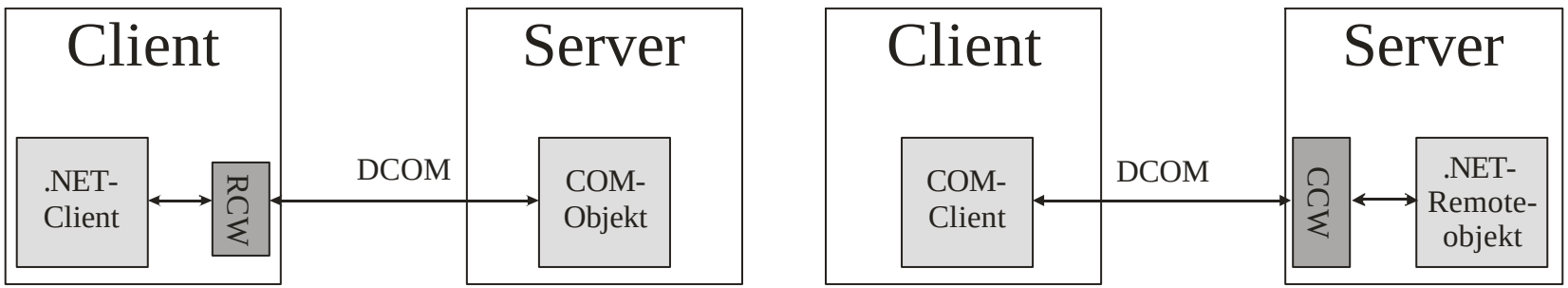
Client	Server	Nutzlast	Protokoll
.NET-Komponente	.NET-Komponente	SOAP/XML	HTTP
.NET-Komponente	.NET-Komponente	Binär	TCP
Verwaltet/nicht verwaltet	.NET-Webdienst	SOAP/XML	HTTP
.NET-Komponente	Nicht verwaltete herkömmliche COM-Komponente	NDR (Network Data Representation, Netzwerkdatendarstellung)	DCOM
Nicht verwaltete herkömmliche COM-Komponente	.NET-Komponente	NDR	DCOM

.NET-Remoting 8/9

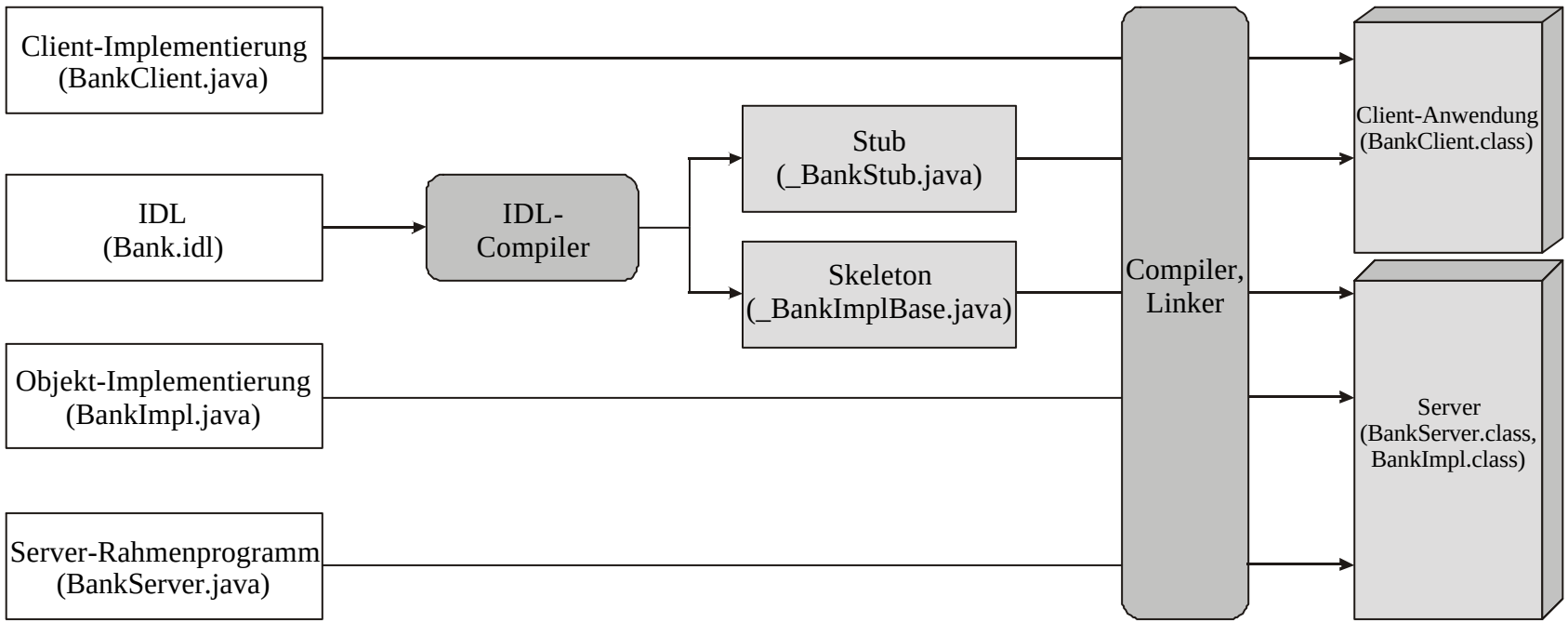


.NET-Remoting 9/9

- Nutzung von COM-Komponenten durch .NET-Anwendungen über Runtime Callable Wrapper
- Nutzung von .NET-Komponenten durch COM-Anwendungen über COM Callable Wrapper



CORBA-Anwendung



Literatur 1/4

- Born, A.: Dot-Net: Bauplatz für Webservices – Alles wird .Net, Network World 21/01
- Breymann, U., Loviscach, J.: Die neue C-Klasse – C# im Vergleich mit C++ und Java, c't 04/02, S. 98-105
- Brien, C.: SOAP – Vergleich mit herkömmlichen VA-Technologien, Hauptseminararbeit, Technische Universität Ilmenau 2002
- Eller, F.: C# lernen, 1. Auflage, Addison Wesley Verlag München 2001
- Groß, R.: Datenaustausch in verteilten Umgebungen – Daily SOAP, Windows 2000 Magazin 03/02

Literatur 2/4

- Gruhn, V., Thiel, A.: Komponentenmodelle, 1. Auflage, Addison Wesley Verlag München 2000
- Klöpper, M.: Aktuelle Initiativen großer Hersteller: Microsoft .NET, Seminararbeit, Universität Münster 2001
- Loviscach, J., Schulz, H., Violka, K.: Sunspiration - .NET und SunONE im Plattformvergleich, c't 04/02, S. 92-97
- Obermeyer, P., Hawkins, J.: Microsoft .NET Remoting: Ein technischer Überblick
- Obermeyer, P., Hawkins, J.: Format for .NET Remoting Configuration Files, MSDN 2001/2002

Literatur 3/4

- Schlede, F.-M.: Interview zu .NET und Microsoft Server – Basis der Visionen, Windows 2000 Magazin 03/02
- Schmidt, H.: Web-Services: Sun und Microsoft konkurrieren um den neuen Megatrend im Internet, Frankfurter Allgemeine Zeitung 38/2002, S. 23
- Siering, P.: Das Microsoft-Internet - .NET und was dranhängt, c't 04/02, S. 86-91
- Srinivasan, P.: Einführung in das Microsoft .NET-Remotingframework, MSDN
- Stal, M.: "Microsoft .NET" – Evolution oder Revolution?, Objektspektrum 06/2000, S. 14-18

Literatur 4/4

- Vawter, C., Roman, E.: J2EE vs. Microsoft.NET – A comparison of building XML-based web services, Sun 2001
- Willers, M.: Microsoft.NET – Alles wird gut!?, MSDN
- Willers, M.: Die ".NET Common Language Runtime": Überblick und technischer Einstieg, Objektspektrum .../2001, S. 91-96