

## Kapitel 1

# Einführung

### **In diesem Kapitel:**

|  |    |
|--|----|
| Was ist .NET?  | 42 |
| Bausteine des .NET Framework                                   | 48 |
| Standardisierung bei ECMA und ISO                              | 54 |
| Plattformen  | 55 |
| Geschichte und Versionen                                       | 59 |
| Produkte und Installationspakete                               | 64 |
| Entscheidung zwischen der deutschen und der englischen Version | 70 |
| Installation   | 74 |

## Was ist .NET?

.NET (gesprochen DOTNET) ist der Oberbegriff für die wichtigste Softwareentwicklungsinfrastruktur der Firma Microsoft. .NET ist heute neben Java die wichtigste Softwareentwicklungsplattform.

**HINWEIS** Bezüglich der genauen Schreibweise von .NET gibt es einige Meinungsverschiedenheiten. Zum Teil schreibt Microsoft selbst *.net* oder *.Net*. Einige Medien schreiben den Begriff aus: *DOTNET* oder *dotnet*. Vorherrschend und von Microsoft selbst meist verwendet ist jedoch die Schreibweise mit drei Großbuchstaben. Diese Schreibweise wird auch in diesem Buch verwendet (außer auf dem Cover, da hier das offizielle Logo abgebildet ist, das die Kleinschreibweise verwendet).

## Definition

.NET ist eine betriebssystemunabhängige Softwareentwicklungsplattform mit Unterstützung für die Programmierparadigmen

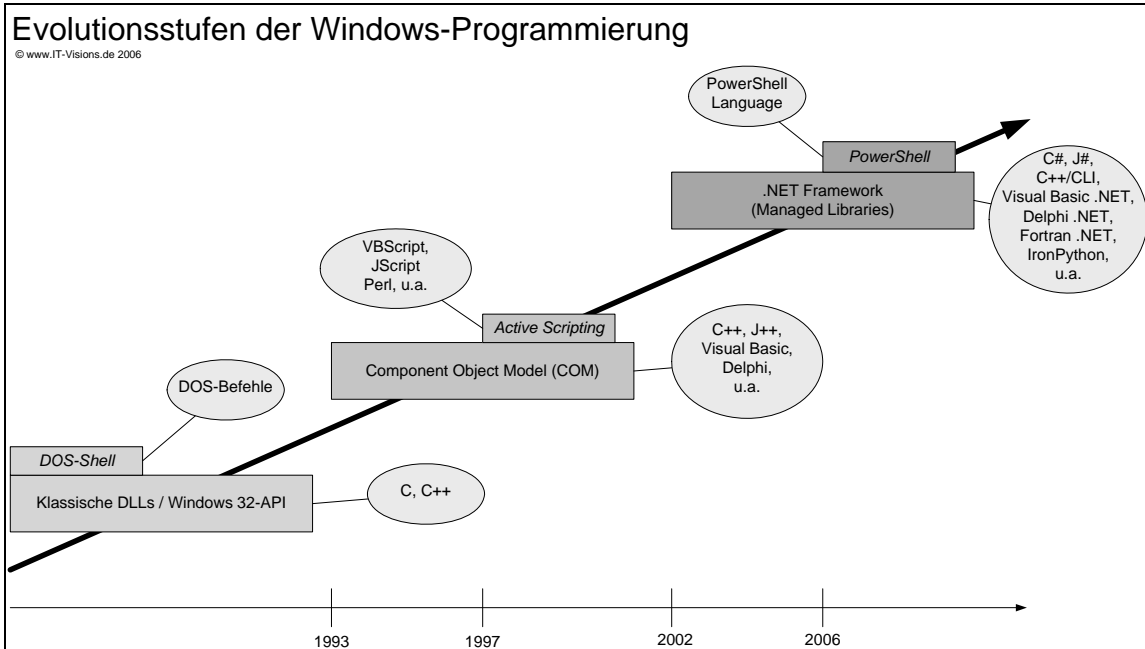
- Objektorientierung
- Komponentenorientierung und
- Serviceorientierung

Microsoft hatte .NET von Anfang an plattformneutral konzipiert, selbst aber keine Bestrebungen zur Implementierung auf Mac und Unix/Linux unternommen. Weite Teile von .NET sind aber inzwischen durch die Initiative anderer Unternehmen (insbesondere Novell) auch für andere Betriebssysteme verfügbar. Und dies wird von Microsoft sogar unterstützt mit Hinblick auf die wachsende Anzahl von Konkurrenzbetriebssystemen, insbesondere im Markt der mobilen Geräte. Microsoft selbst bietet mit Silverlight eine Variante von .NET auch für Mac OS an.

## Ziele von .NET

Primäres Ziel bei der Entwicklung von .NET war es, eine moderne, konsistente, flexible und sichere Softwareentwicklungsplattform auf hohem Abstraktionsniveau für die Entwicklung von Software jeder Art zu schaffen. Zuvor war die Anwendungsentwicklung in verschiedenen Programmiersprachen sehr uneinheitlich und unterschiedlich mächtig. Diese Unterschiede will .NET beseitigen.

Die .NET-Plattform ist insofern vollständig, als schon heute alle Arten von Windows-Anwendungen (mit Ausnahme von Hardware-Treibern) damit entwickelt werden können. Die mitgelieferte Klassenbibliothek ist bereits sehr umfangreich und deckt viele Funktionen ab. Natürlich gibt es weiterhin auch Programmierschnittstellen, die nur in klassischer Form (C-API oder COM-DLL) vorliegen.



**Abbildung 1.1** .NET ist die dritte Evolutionsstufe der Windows-Programmierung

## Vielfalt von .NET

Die Vielfalt der .NET-Plattform drückt das folgende Bild aus. Für alle Arten von Anwendungen (Web, Desktop, Geräte) bietet das .NET Framework inzwischen mehr als eine Alternative. Zudem gibt es Alternativen für Datenzugriffe, die Implementierung von Diensten sowie zahlreiche Werkzeuge. Dabei nicht berücksichtigt sind die vielen Drittanbieter und Open Source-Projekte, die weitere Alternativen ergänzen.

In allen Fällen stehen mit C# und Visual Basic die gleichen Programmiersprachen, mit Visual Studio das gleiche Werkzeug und den .NET-Bibliotheken die gleichen Klassen zur Verfügung.

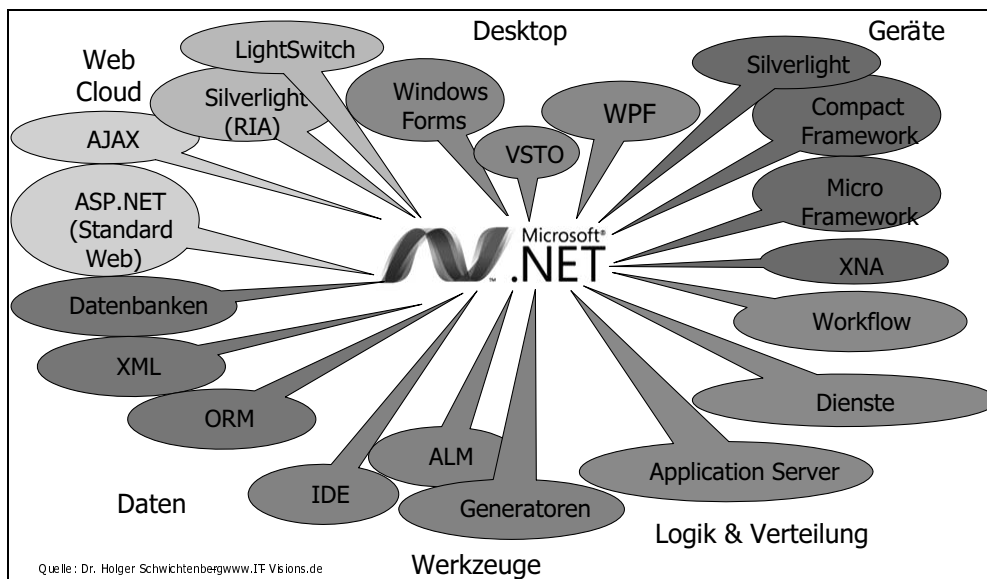


Abbildung 1.2 .NET als durchgängige Plattform

**HINWEIS**

Nicht Ziel der .NET-Plattform ist die hardwarenahe Programmierung, zum Beispiel die Entwicklung von Treibern.

.NET hat natürlich viele Anleihen bei Java genommen, sowohl bei der Sprachsyntax als auch den Klassenbibliotheken. Aber auch Java hat inzwischen Einiges bei .NET abgeschaut. Als Beispiele sind Annotationen, Autoboxing und Generics ab Java 5 zu nennen. Bei den im Standard mitgelieferten Klassen haben beide Plattformen inzwischen die 10.000er Marke gerissen. Jede konkrete Java EE-Implementierung fällt noch weitaus größer aus. Darüber hinaus gibt es unzählige Zusatzbibliotheken in beiden Welten. Viele .NET-Entwickler sahen einen Vorteil von .NET gegenüber Java darin, dass »alles« von Microsoft aus einer Hand geliefert wird, während Java-Entwickler die Qual der Wahl vieler (miteinander konkurrierender) Open Source-Bibliotheken haben und meist diese Wahl auch lieben.

Inzwischen ist die .NET-Welt auch nicht mehr so einfach. Neben einer Vielzahl an Drittanbieter- und Open Source-Bibliotheken (siehe SourceForge <http://sourceforge.net/> und Microsofts eigenes Quellcodeportal CodePlex <http://www.codeplex.com/>) bringt auch Microsoft ständig neue .NET-Bibliotheken heraus, die weitere Abstraktionen für schon vorhandene Funktionen bieten oder kommende Erweiterungen von .NET vorab liefern. Das Tempo der An- und auch Absagen von Bibliotheken ist bei Microsoft inzwischen derart atemberaubend, dass sich mancher .NET-Entwickler vorkommt wie auf dem Java-Basar, den er vermeiden wollte. Microsoft liefert immer häufiger Bibliotheken auch als Open Source-Code, manchmal sogar dennoch mit offiziellem Produktsupport. Trotz der inzwischen zahlreichen Newsgroups, Mailinglisten und Webforen gibt es viele .NET-Entwickler, die sich gerade deshalb für .NET entscheiden, weil sie kommerziellen Support aus einer Hand erhalten, der bei Fehlern und unerwartetem Verhalten die »Schuld« nicht so leicht auf andere Software abwälzen kann.

## Verbreitung und Akzeptanz von .NET

Die Akzeptanz von .NET verlief in Deutschland sehr schleppend. Bezeichnend der Wikipedia-Eintrag zum Stand von .NET im Jahre 2004: »Obwohl technisch schon einige Jahre alt, steht der Marktanteil an .NET-Programmen immer noch in keinem Verhältnis zur Aufmerksamkeit in Medien und Entwicklergemeinde« (inzwischen gibt es in [WP01] diesen Satz nicht mehr).

Dazu beigetragen haben auch zwei Fehler bei Microsoft:

- In den Jahren 2001 bis 2004 hatte Microsoft *.NET* zunächst als Marketing-Begriff für alle neuen Produkte (Betriebssystem, Server, Office) verwendet – nach Kritik von Kunden und Medien hat die Firma jedoch die sinnvolle Reduzierung auf das *.NET Framework* und die zugehörigen Softwarekomponenten und Werkzeuge vollzogen
- Microsoft hat lange Zeit die eigene Produktentwicklung nicht auf .NET umgestellt. Zeitweise hatte Microsoft daher bei Kunden ein Glaubwürdigkeitsproblem: Man fragt sich, warum man .NET einsetzen soll, wenn bei Microsoft Entwicklungsabteilungen selbst immer noch mit klassischem C++ entwickeln.

Auch in dem aktuellen Windows-Client-Betriebssystem *Windows 7* ist kein einziger Teil mit dem .NET Framework geschrieben worden. Es gilt lediglich, dass .NET 3.5 Service Pack 1 mit Windows 7 ausgeliefert wird und man .NET 4.0 zusätzlich installieren kann.

Anders sieht es in Windows Server 2008 R2 aus. Dort sind dies zum Beispiel PowerShell, das PowerShell Integrated Scripting Environment (ISE), die Gruppenrichtlinienverwaltungskonsolle und die Teile anderer MMC-Konsolen. Positive Beispiele für den Einsatz von .NET im Hause Microsoft sind allerdings Microsoft SharePoint und Microsoft CRM. Auch die aktuelle Version Visual Studio 2010 ist zum Teil in .NET-Code reimplementiert worden.

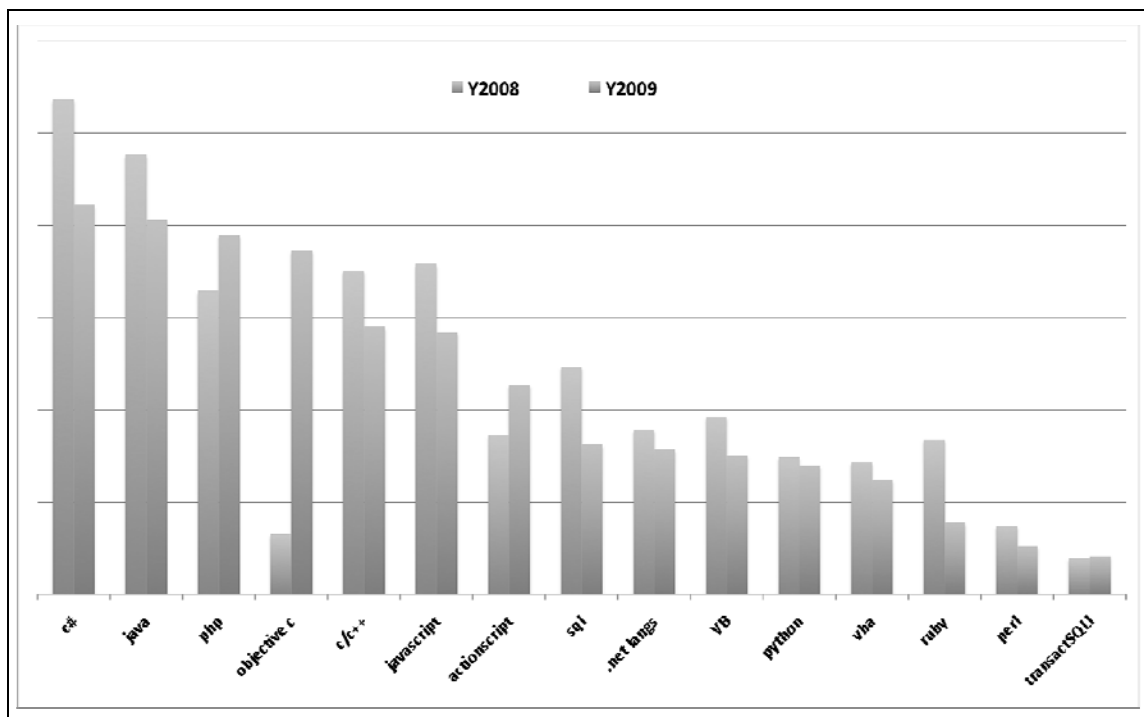
Seit der Einführung von .NET 2.0 im November 2005 verzeichnen Schulungsunternehmen, Berater und Softwarehäuser jedoch einen wahren Boom in Hinblick auf .NET. Mittlerweile werden erfahrene .NET-Softwareentwickler händeringend gesucht. Die Adaption von .NET 3.0 hingegen vollzog sich wieder viel langsamer. Ein Grund dafür war, dass .NET 3.0 schon ein Jahr nach .NET 2.0 erschienen ist. Ein weiterer Grund war, dass .NET 3.0 in wesentlichen Teilen eine Alternative für Technologien in .NET 2.0 war und ein Umstieg auf die neuen Technologien beschwerlich war. Mit .NET 3.5 bzw .NET 3.5 Service Pack gibt es – insbesondere durch die dort eingeführten Technologien zum Datenzugriff – mehr Gründe für einen Versionswechsel. .NET 4.0 liefert anders als .NET 3.0 und .NET 3.5 keine großen neuen Bibliotheken, sondern kleinere Ergänzungsbibliotheken sowie Verbesserungen für die bestehenden Bibliotheken. Daher wird .NET 4.0 im Markt als eine Abrundung von .NET angesehen.

Es gibt keine verlässlichen Marktdaten über die Verbreitung von .NET. Es gibt verschiedene Statistiken, die man als Indiz heranziehen könnte, z.B. den GULP.de Projektmarktindex (siehe Bildschirmabbildung).

| Programmiersprachen |            |             |                   |                         |
|---------------------|------------|-------------|-------------------|-------------------------|
| Platz               | Skill      | Ø-Std. Satz | Angebote Projekte | verfügbare Freiberufler |
| 1. ( 1 )            | Java       | 69 EUR      | 15,7% ( 15,4% ) ➡ | 734 von 6384            |
| 2. ( 2 )            | SQL        | 65 EUR      | 12,9% ( 12,1% ) ↗ | 527 von 4353            |
| 3. ( 3 )            | C/C++      | 66 EUR      | 9,2% ( 10,7% ) ↘  | 684 von 5356            |
| 4. ( 4 )            | ABAP4      | 80 EUR      | 6,1% ( 5,8% ) ➡   | 226 von 1154            |
| 5. ( 5 )            | JavaScript | 58 EUR      | 4,0% ( 4,0% ) ➡   | 82 von 850              |
| 6. ( 6 )            | C#         | 64 EUR      | 3,6% ( 3,8% ) ➡   | 295 von 1746            |
| 7. ( 8 )            | PL/SQL     | 71 EUR      | 3,0% ( 2,6% ) ↗   | 120 von 588             |
| 8. ( 7 )            | Basic      | 64 EUR      | 2,5% ( 2,6% ) ↘   | 155 von 1817            |
| 9. ( 9 )            | Perl       | 67 EUR      | 1,7% ( 2,0% ) ↘   | 68 von 805              |
| 10. ( 10 )          | PHP        | 59 EUR      | 1,7% ( 1,6% ) ➡   | 187 von 2519            |

**Abbildung 1.3** GULP.de-Freiberufler-Projektmarktindex der Nachfrage nach Qualifikationen und gebotener Stundensätze (»Glupometer«) [GU01]

Ein anderer Indikator wären die Verkäufe von Computerbüchern. O'Reilly erhebt eine solche Statistik verlagsübergreifend für die USA. Dabei liegt C# allein schon vor Java und inklusive *.NET Languages* und VB sehr deutlich vor Java.



**Abbildung 1.4** Relativer Vergleich der Anzahl verkaufter Bücher in den USA – 2008 und 2009. Quelle: [OR01]

## Technische Merkmale des .NET Framework

Wesentliche Merkmale des .NET Framework sind:

- Parallelbetrieb verschiedener .NET Framework-Versionen (eine .NET-Anwendung startet automatisch mit der Framework-Version, für die sie entwickelt wurde)
- durchgängige Objektorientierung: auch elementare Datentypen wie Zahlen und Zeichenketten sind Objekte
- wiederverwendbare Softwarekomponenten (sogenannte Assemblys)
- Plattformunabhängigkeit durch Zwischensprache/Intermediation mit Just-in-Time-Compiler wie bei Java: Write Once Run Anywhere (WORA)
- verschiedene Anwendungstypen (*Fat-Clients*, Standard-Webanwendungen, Rich Internet Applications, Systemdienste, Webdienste, Pocket-PC-Anwendungen, SmartPhone-Anwendungen)
- Sprachunabhängigkeit (mehr als 45 verschiedene Programmiersprachen) mit sprachübergreifenden Aufrufen und sprachübergreifender Vererbung
- einheitliche Laufzeitumgebung mit Diensten wie Codeüberprüfung (Sicherheit, Array-Grenzen etc.), Threading, Speicherbereinigung und Ausnahmebehandlung
- umfangreiche Klassenbibliothek mit mehr als 9.000 Klassen, einheitlich für alle .NET-fähigen Programmiersprachen
- Schnittstellenverträge, die ermöglichen, dass man Mitglieder ergänzt, ohne den Schnittstellenvertrag zu brechen (Der Vertrag wird erst gebrochen, wenn man Mitglieder oder Parameter entfernt bzw. Datentypen ändert.)
- XML-basierte Konfiguration von Anwendungen (Abkehr von der Windows-Registrierungsdatenbank)
- Schutz vor »gefährlichem« Programmcode durch Sandbox-Konzepte wie in Java
- Nutzung von Metadaten (automatische Metadatengenerierung und manuelle Metadaten)
- XCopy-Deployment (Verteilung von Anwendungen durch einfaches Kopieren der Programmdateien sowie der zugehörigen Bibliotheken und Ressourcendateien)
- Interoperabilität zu Plattformen: COM, Windows 32 API sowie XML-Webservices (zu Java und CORBA auch durch Drittanbieterprodukte)

## Bausteine des .NET Framework

Das .NET Framework ist ein sehr komplexes Gebilde, das durch einige Zusatzprodukte (z.B. Microsoft BizTalk, Microsoft SharePoint) noch umfangreicher wird. Im Folgenden sollen drei grafische Darstellungen aus verschiedenen Blickwinkeln einen Überblick über das .NET Framework geben.

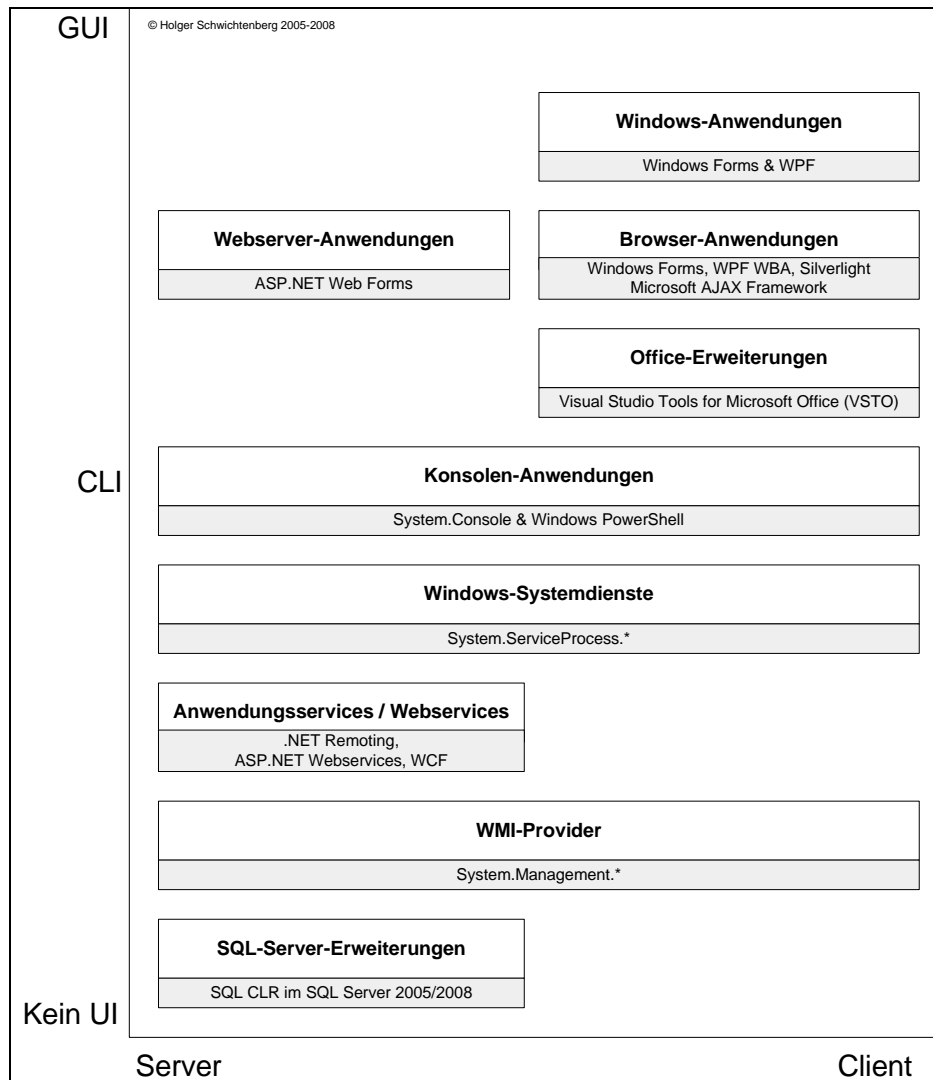
### Blickwinkel 1: Anwendungstypen

Eine Möglichkeit der Gliederung des .NET Framework sind Anwendungstypen. Das .NET Framework unterstützt sowohl Client- als auch Server-Szenarien durch verschiedene Anwendungstypen, die im Folgenden aufgelistet sind:

- Windows-Anwendungen auf Basis von Windows Forms oder Windows Presentation Foundation (WPF) oder Microsoft Silverlight in der *Out-Of-Browser (OOB)*-Variante
- Webserver-Anwendungen auf Basis von ASP.NET Webforms, ASP.NET Modell View Controller und ASP.NET Dynamic Data
- Web-Client-Anwendungen auf Basis von ASP.NET AJAX (HTML/JavaScript ohne Plug-Ins) oder mit Plug-Ins durch Einsatz von WPF im Browser (*Web Browser Application – WBA* alias *XBAP*) oder Microsoft Silverlight. (Ab .NET 4.0 wird Windows Forms im Browser mit IEHost.dll nicht mehr unterstützt.)
- Erweiterungen für Microsoft Office auf Basis der Visual Studio Tools for Microsoft Office (VSTO)
- Konsolenanwendungen auf Basis der .NET-Klasse `System.Console`
- Windows-Dienste auf Basis der Klassen in der Komponente `System.ServiceProcess`
- XML-Webservices auf Basis von ASP.NET Webservices
- WMI-Provider auf Basis der Klassen in `System.Management`
- Prozeduren, Funktionen und Datentypen für den Microsoft SQL Server 2005/2008 (inkl. R2) in Form der *SQLCLR*

Die folgende Grafik verdeutlicht den Sachverhalt in einem zweidimensionalen Raster, bei dem auf der X-Achse die Entscheidung zwischen Client und Server und auf der Y-Achse die Frage aufgetragen ist, ob es eine grafische Benutzerschnittstelle (Graphical User Interface – GUI), eine Kommandozeilenschnittstelle (Command Line Interface – CLI) oder gar keine Benutzerschnittstelle (User Interface – UI) gibt.





**Abbildung 1.5** Anwendungstypen im .NET Framework und deren typische Rollen

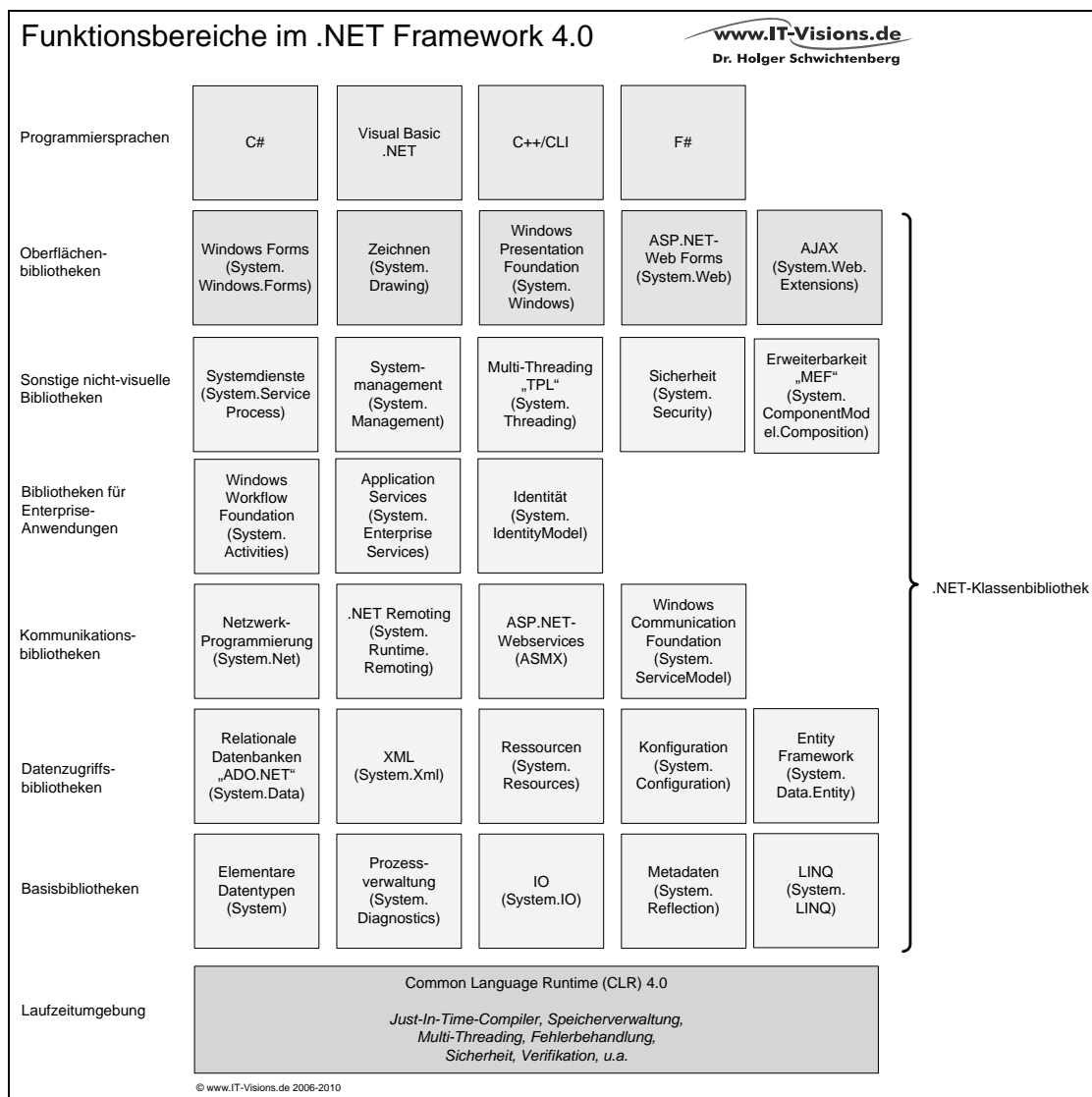
## Blickwinkel 2: Funktionsbereiche

Die zweite Abbildung zeigt die Bausteine nach Funktionsbereichen. Die Grafik gliedert sich in:

- Laufzeitumgebung (Common Language Runtime, CLR),
- darauf aufbauende Bibliotheken wie System.IO, System.Data (alias ADO.NET), System.Management, System.Workflow, System.Reflection, System.Xml u.a.

- Oberflächenbibliotheken wie System.Windows.Forms (alias Windows Forms), System.Windows (alias Windows Presentation Foundation) und System.Web (alias ASP.NET)
- Sprach-Compiler für Visual Basic, C#, C++/CLI, F# u.a.

**HINWEIS** Die Oberflächenbibliothek gehört mit zur sogenannten *.NET-Klassenbibliothek*. Auch die Sprachcompiler könnte man mit dazu zählen, da man die Kompilierung auch über die .NET-Klassenbibliothek steuern kann.



**Abbildung 1.6** Zentrale Bausteine der .NET-Strategie

## Blickwinkel 3: Aufteilung nach Schichten

Eine dritte alternative Darstellung der .NET-Technologien nach den Schichten im klassischen Dreischichtmodell (Benutzerschnittstelle, Logik und Datenzugriff) zeigt die folgende Abbildung. Anschließend gibt es kurze Erläuterungen zu der Grafik.

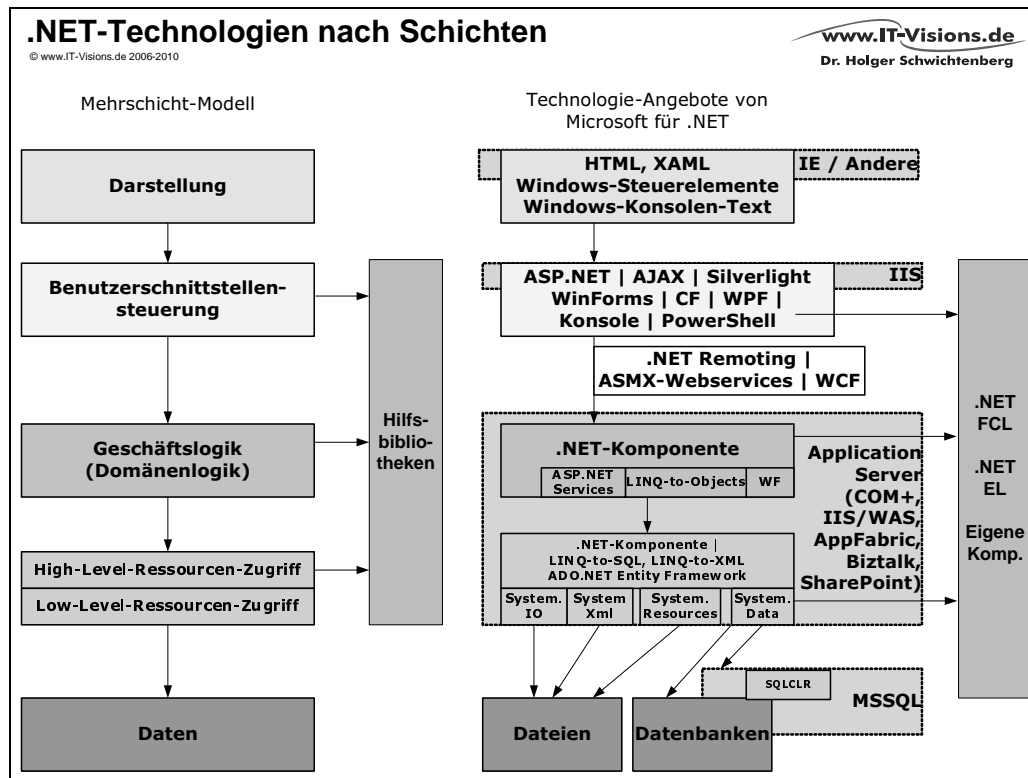


Abbildung 1.7 Die Einordnung der .NET-Technologien in das Modell der mehrschichtigen Anwendungsentwicklung

### HINWEIS Mehrschichtige .NET-Anwendungen versus RAD

.NET erlaubt sowohl monolithisches Entwickeln (Oberfläche, Logik und Datenzugriff in einer Assembly) im Sinne des Rapid Application Development (RAD) als auch das mehrschichtige Entwickeln (oft auch als *Enterprise Development* bezeichnet).

RAD-Anwendungen entstehen oft durch den Einsatz von Assistenten und Ziehen/Fallenlassen innerhalb einer grafischen Entwicklungsumgebung. Ein Entwickler kann so in wenigen Minuten umfangreiche Anwendungen »zusammenklicken«, für die er mit »richtigem« Programmieren mehrere Stunden oder Tage benötigt hätte. Leider ist das Ergebnis von RAD in den seltensten Fällen wieder verwendbar, wartbar, skalierbar oder sicher. Ein typisches Merkmal von RAD ist die enge Vermischung von Darstellung, Logik und Datenzugriff.

Microsoft hat in den letzten Jahren sehr viel Wert auf das Mehrschichtmodell gelegt und dabei zum Teil Entwickler vernachlässigt, die sich an extremem RAD im Sinne von Microsoft Access, Visual Basic 6.0 oder Visual FoxPro begeistern können. Mit *LightSwitch* (erste Beta erschienen im August 2010, daher nicht Thema dieses Buchs) versucht Microsoft, RAD und Mehrschichtigkeit zu verheiraten.

## Benutzerschnittstellensteuerung

Im Bereich der Benutzerschnittstellensteuerung unterstützt .NET 4.0:

- Desktop-Anwendungen durch Windows Forms und Windows Presentation Foundation (WPF)
- Webanwendungen durch ASP.NET, die ASP.NET-AJAX-Erweiterungen und Silverlight (Windows Forms und WPF können auch im Browser ablaufen, dies wird jedoch selten verwendet)
- Anwendungen für mobile Endgeräte durch Windows Forms im .NET Compact Framework (CF) bzw. Silverlight und XNA (ab Windows Phone 7)
- Konsolenanwendungen durch die Klasse `System.Console` sowie die Windows PowerShell

Für die Desktop-Anwendungen zeichnet sich auf der .NET-Plattform ein ähnlicher Konkurrenzkampf wie bei Java mit AWT, SWT und Swing ab. WPF bietet mehr grafische Möglichkeiten als Windows Forms, es fehlen aber noch Steuerelemente und die Werkzeuge sind noch nicht so ausgereift.

Im Bereich der Webentwicklung bietet Microsoft mit ASP.NET einen hohen Abstraktionslevel. Die Oberflächenbeschreibung in ASP.NET erfolgt durch HTML (Hypertext Markup Language), eingebettete XML-Fragmente (Extensible Markup Language) und Programmcode. XML und Programmcode erzeugen zusammen (browserunabhängiges) HTML, CSS und JavaScript. Durch die im Januar 2007 erschienenen AJAX-Erweiterungen ist ASP.NET nun nicht mehr als reines Server-Framework zu bezeichnen. Alle Client-Funktionen von ASP.NET basieren auf JavaScript. Es wird weder ActiveX noch eine andere Komponententechnologie verwendet, die eine spezielle Laufzeitumgebung auf dem Client erfordert. Browserbasierte Anwendungen sind auch möglich auf Basis von Windows Forms und WPF, machen dann aber Internet Explorer und .NET Framework auf dem Client erforderlich (seit .NET 3.5 läuft WPF auch im Firefox).

Microsoft Silverlight ist ein Kompromiss: Silverlight braucht eine sehr kleine .NET-Laufzeitumgebung und läuft in verschiedenen Browsern und verschiedenen Betriebssystemen (Microsoft unterstützt Windows und Mac OS, Mono auch andere).

## Geschäftslogik

Softwarekomponenten, die in der Geschäftslogik und beim Datenzugriff zum Einsatz kommen, sind grundsätzlich normale .NET-Assemblys, unabhängig davon, ob sie im Anwendungsserver gehostet werden oder Teile des eigentlichen Anwendungsprozesses sind.

Auf der Ebene der Geschäftslogik bietet Microsoft bisher recht wenig Unterstützung. Seit .NET 2.0 gibt es verschiedene Anwendungsdienste in Webanwendungen (z.B. Benutzerverwaltung, Rollenverwaltung und Profildatenspeicherung), seit .NET 3.0 die Windows Workflow Foundation (WF) und seit .NET 3.5 auch LINQ to Objects.

Häufig verweist Microsoft in Hinblick auf Geschäftslogik auf den Microsoft BizTalk Server. BizTalk Server ist ein Application Server für Enterprise Application Integration (EAI), Business Process Management (BPM) und die Realisierung eines Enterprise Service Bus (ESB). Die Hyper-Wörter lassen schon vermuten: BizTalk ist nur für große Lösungen geeignet. Die Anschaffung und der Betrieb sind kostenintensiv.

## Ressourcenzugriff

Beim Datenzugriff unterstützt .NET sowohl relationale Datenbanken als auch XML. Beide Welten sind in .NET enger verknüpft, als man dies aus vielen anderen Umgebungen kennt. Ein Entwickler kann relationale Daten mit einem Einzeiler in XML umwandeln oder XML-Daten (mit bestimmten Einschränkungen) ebenso einfach in eine verknüpfte Menge von Tabellen überführen. ADO.NET (Namensraum `System.Data`) und XML.NET (Namensraum `System.Xml`) sind die relevanten Bausteine. XML.NET unterstützt neben den W3C-Standards für XML, XML DOM, XSD, XPATH und XSLT auch einige Microsoft-eigene (aber meist schnellere) Zugriffsverfahren auf XML-Dokumente. ADO.NET hat nur noch eingeschränkt Ähnlichkeit mit dem Vorgänger ADO, denn auch hier verzichtet Microsoft auf die enge Kopplung und setzt daher in der ganzen Datenzugriffsarchitektur auf das optimistische Sperren. Datenänderungskonflikte werden nicht im Vorhinein durch Sperren verhindert: Zur Vermeidung von Deadlocks und Skalierbarkeitsproblemen darf man immer lesen und zurückschreiben, sodass (automatisch oder manuell zu lösende) Änderungskonflikte auftreten können.

Mit .NET 3.5 wurde auch Objektrelationales Mapping (ORM) mit LINQ toSQL und LINQ toEntities (alias ADO.NET Entity Framework, ab Service Pack 1) eingeführt.

Neu seit .NET 2.0 ist die Möglichkeit, .NET-Code auch direkt innerhalb des Microsoft SQL Server-Datenbankmanagementsystems auszuführen. Hier spricht Microsoft von *SQLCLR* oder *CLR-Integration*. Dieses Thema kann allerdings aus Platzgründen in diesem Buch gar nicht behandelt werden.

## Verteilte .NET-Anwendungen

Der Aufruf von entferntem Code via Remote Procedure Call (RPC) oder über das auf dem RPC aufsetzende Distributed COM (DCOM) war aufgrund der Komplexität und der mangelnden Durchlässigkeit bei Firewalls bislang sehr unbefriedigend. .NET Framework 4.0 kennt für den Aufruf von Code in anderen Application Domains, anderen Prozessen oder auf anderen Systemen unterschiedliche Fernaufrufkonzepte mit verschiedenen Serialisierungsformaten, Transportwegen und Anwendungsgebieten:

- ASP.NET-basierte XML-Webservices (ASMX) – veraltet
- .NET Remoting – veraltet
- Windows Communication Foundation (WCF)

Mit .NET 1.0 gab es bereits zwei Konzepte für den Aufruf von Code in anderen Prozessen oder auf anderen Systemen: die auf W3C-Standards basierenden ASP.NET Webservices (ASMX) und das proprietäre (aber schnellere) .NET Remoting. WCF ist eine Zusammenführung beider Welten, wobei aber auch einige Spielarten der Kommunikation ausgeblendet werden. WCF favorisiert ein loses Kopplungsmodell. Die enge Kopplung von .NET Remoting, die man auch von CORBA und Java RMI kennt, ist in .NET zum Aussterben verurteilt worden.

---

**HINWEIS** .NET unterstützt auch die Kommunikation mit Low-Level-Konzepten wie

- Direkte TCP/IP-Kommunikation bzw. Anwendungsprotokolle wie HTTP oder FTP
- Named Pipes
- Memory Mapped Files

Grundsätzlich können Sie auch unter .NET weiterhin Distributed COM als Fernaufrufprotokoll nutzen. DCOM gilt jedoch als veraltet und sollte allenfalls noch zur Kommunikation mit alten DCOM-Gegenstellen verwendet werden.

---

## Application Server

Ein Application Server dient der Bereitstellung von Diensten (Services) für den entfernten Aufruf. In der .NET-Welt war das Thema Application Server jahrelang ein Politikum. Nachdem es in der COM-Welt vor .NET mit COM+ noch ein Äquivalent zum JEE-Application Server-Konzept gab, versuchte Microsoft erst sein altes COM+ unter dem neuen Namen *.NET Enterprise Services* (System.EnterpriseServices) zu verkaufen und dann war Microsoft lange Jahre der Meinung, dass es eines dedizierten Application Servers für .NET nicht bedarf. Interessanterweise gab es hier auch keine nennenswerten Versuche von Drittanbietern, um diese zu Lücke zu schließen.

Microsoft pries den Internet Information Server (IIS) oft als *Application Server* an. Aber lange Zeit verstand der IIS nur das HTTP-Protokoll. Seit Version 7.0 kann er auch andere Protokolle wie TCP, MSMQ und Named Pipes hosten (*Windows (Process) Activation Service* (WAS)). Aber es fehlten ihm weiterhin Werkzeuge für die Installation, Verwaltung und Überwachung von Diensten.

Da der Microsoft SharePoint Server auf dem IIS/WAS basiert, kann man auch SharePoint als Application Server für .NET-Dienste einsetzen.

Dies hat Microsoft nun Mitte 2010 mit *Windows Server AppFabric* ergänzt. Windows Server AppFabric ist eine kostenfreie Erweiterung für Windows Server. Die Verwaltung erfolgt über eine Erweiterung der IIS-Verwaltungskonsolle. AppFabric stellt auch einen eigenen Dienst bereit: das verteilte Zwischenspeichern (Caching).

## Orthogonale Bibliotheken

Als *Orthogonale Bibliotheken* bezeichnet man solche, die man auf jeder Schicht im Schichtenmodell verwenden kann. Dazu gehören die meisten Klassen der .NET-Klassenbibliothek z.B. für elementare Datentypen, Objektmengen, Textbearbeitung und Multi-Threading. Außerdem gehören dazu Zusatzbibliotheken wie die .NET Enterprise Library (.NET EL).

# Standardisierung bei ECMA und ISO

Microsoft hat einige Teile des .NET Framework unter dem Namen *Common Language Infrastructure* (CLI) standardisieren lassen. Die CLI wurde erstmals im Dezember 2001 von der European Computer Manufacturers Association (ECMA) standardisiert (ECMA-Standard 335, Arbeitsgruppe TC49/TG3, früher: TC39/TG3, siehe [ECMA01]); mit kleinen Änderungen wurde der Standard im Dezember 2002 von der weltweit wichtigsten Standardisierungsorganisation, der International Standardization Organization (ISO), übernommen als ISO/IEC 23271.

Auch die Programmiersprache C# ist von beiden Gremien akzeptiert (ECMA-334 bzw. ISO/IEC 23270). Die Begriffe lauten in den Standards zum Teil allerdings anders als bei Microsoft: Was im .NET Framework *Microsoft Intermediate Language* (MSIL) heißt, entspricht im Standard der *Common Intermediate Language* (CIL). Anstelle der *Framework Class Library* (FCL) spricht man von der *CLI Class Library*. Von der Standardisierung ausgenommen sind jedoch z.B. die Datenbankschnittstelle ADO.NET und die Benutzeroberflächen-Bibliotheken Windows Forms und ASP.NET Webforms. Auch die neueren .NET-Bibliotheken (WPF, WCF und WF) sind nicht standardisiert.

Die Standardisierung der erweiterten Syntax von C# 2005 sowie der Neuerungen in der Common Language Runtime (CLR) 2.0 bei ECMA und ISO sind inzwischen ebenfalls abgeschlossen. Die ECMA hat im Juni 2005 bereits die dritte Version der Standards für C# (ECMA-Standard 334, Arbeitsgruppe TC49/TG2, früher: TC39/TG2) und der Common Language Infrastructure (CLI) (ECMA-Standard 335, Arbeitsgruppe TC49/TG3, früher: TC39/TG3) beendet. Es handelt sich deshalb um die dritte und nicht erst um die zweite Version, weil die ECMA nach dem ursprünglichen Standardisierungsprozess um eine Ratifizierung durch die International Standardization Organization (ISO) gebeten hatte. Die von der ISO geforderten Nachbesserungen führten zur *Second Edition*. Im Rahmen der Ratifizierung durch die ISO wurde dann abermals eine neue Version veröffentlicht.

Der endgültige Standard zu .NET 2.0 ist ECMA-335, 4. Ausgabe Juni 2006 sowie ISO/IEC 23271:2006 (Stand 29.07.2006). C# 2.0 ist standardisiert als ECMA-334, 4. Ausgabe Juni 2006 sowie ISO/IEC 23270:2006 (Stand 23.08.2006).

Ein weiterer, von Microsoft initiiertes Standard ist von der ECMA im Dezember 2005 unter ECMA-372 (Arbeitsgruppe TC49/TG5, früher: TC39/TG5) verabschiedet worden: C++/CLI ist eine Spracherweiterung für C++ (ISO/IEC 14882:2003), die eine elegantere Nutzung von C++ auf der CLI-Plattform ermöglicht, als dies bisher mit den Managed Extensions for C++ (alias Managed C++) möglich war.

**HINWEIS** Auf der Website der entsprechenden ECMA-Arbeitsgruppen (TC49/TG2, 3 und 5, siehe [ECMA02]) gab es bis zum Redaktionsschluss keine Meldungen über die Standardisierung der Neuerungen in C# 3.0 oder der Klassenbibliotheken in .NET 3.0 und 3.5.

## Plattformen

.NET ist grundsätzlich plattformunabhängig – wie die Programmiersprache Java verwendet .NET eine Zwischensprache. Microsoft selbst bedient primär nur die Windows-Betriebssysteme. MacOS wird nur durch Silverlight von Microsoft unterstützt. Mono ist eine »Nachentwicklung« von .NET für andere Plattformen durch die Firma Novell. Mono hat aber eine andere Versionsnummernzählung.

Die nachstehende Tabelle zeigt die Verfügbarkeit von .NET für verschiedene Betriebssysteme.

|   | .NET 1.0        | .NET 1.1        | .NET 2.0        | .NET 3.0        | .NET 3.5        | .NET 4.0        |
|---|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| <b>Windows 3.1/<br/>Windows NT 3.51</b> | Nicht verfügbar | Nicht verfügbar | Nicht verfügbar | Nicht verfügbar | Nicht verfügbar | Nicht verfügbar |
| <b>Windows 95</b>                       | Nicht verfügbar | Nicht verfügbar | Nicht verfügbar | Nicht verfügbar | Nicht verfügbar | Nicht verfügbar |
| <b>Windows 98</b>                       | Add-On          | Add-On          | Add-On          | Nicht verfügbar | Nicht verfügbar | Nicht verfügbar |
| <b>Windows ME</b>                       | Add-On          | Add-On          | Add-On          | Nicht verfügbar | Nicht verfügbar | Nicht verfügbar |
| <b>NT 4.0</b>                           | Add-On          | Add-On          | Nicht verfügbar | Nicht verfügbar | Nicht verfügbar | Nicht verfügbar |
| <b>Windows 2000</b>                     | Add-On          | Add-On          | Add-On          | Nicht verfügbar | Nicht verfügbar | Nicht verfügbar |
| <b>Windows XP</b>                       | Add-On          | Add-On          | Add-On          | Add-On          | Add-On          | Add-On (SP3) ►  |

|                                  | .NET 1.0                                       | .NET 1.1        | .NET 2.0                                   | .NET 3.0                                   | .NET 3.5                                   | .NET 4.0     |
|----------------------------------|--|-----------------|--|--|--|--------------|
| <b>Windows XP Service Pack 2</b> | Add-On   | Add-On          | Enthalten                                  | Add-On                                     | Add-On                                     | Add-On (SP3) |
| <b>Windows Server 2003</b>       | Add-On   | Add-On          | Add-On                                     | Add-On                                     | Add-On                                     | Add-On (SP2) |
| <b>Windows Server 2003 R2</b>    | Add-On   | Add-On          | Enthalten                                  | Add-On                                     | Add-On                                     | Add-On (SP2) |
| <b>Windows Vista</b>             | Add-On   | Add-On          | Enthalten                                  | Enthalten                                  | Add-On                                     | Add-On       |
| <b>Windows 2008</b>              | Nicht empfohlen, es erscheint eine Warnmeldung | Add-On          | Enthalten                                  | Optionales Betriebssystem-feature          | Add-On                                     | Add-On       |
| <b>Windows Server 2008 Core</b>  | Nicht verfügbar                                | Nicht verfügbar | Nicht verfügbar                            | Nicht verfügbar                            | Nicht verfügbar                            | Nein         |
| <b>Windows 2008 R2</b>           | Nicht empfohlen, es erscheint eine Warnmeldung | Add-On          | Enthalten                                  | Optionales Betriebssystem-feature          | Optionales Betriebssystem-feature          | Add-On       |
| <b>Windows 2008 R2 Core</b>      | Nicht empfohlen, es erscheint eine Warnmeldung | Add-On          | Teilw. – optionales Betriebssystem-feature | Teilw. – optionales Betriebssystem-feature | Teilw. – optionales Betriebssystem-feature | Nein         |
| <b>Windows 7</b>                 | Nicht empfohlen, es erscheint eine Warnmeldung | Add-On          | Enthalten                                  | Enthalten                                  | Enthalten                                  | Add-On       |

Tabelle 1.1 Verfügbarkeit der .NET-Versionen

## Microsoft .NET Framework

Microsoft selbst versteht unter Plattformunabhängigkeit nur seine eigenen Windows-Varianten: DOS-basierte Windows-Systeme (98, ME), NT-basierte Windows-Systeme (NT 4.0, 2000, XP, 2003, Vista, 2008 inkl. R2-Version) sowie CE-basierte Systeme (Pocket PC, Windows Mobile, Windows Phone). Für Windows CE-basierte Systeme gibt es eine reduzierte Version, das sogenannte *.NET Compact Framework*. Silverlight ist ebenfalls eine reduzierte Version von .NET. Für sehr kleine Endgeräte (z.B. Uhren, Hilfsanzeigen von Notebooks) existiert ein .NET Micro Framework (MF). Für die Programmierung der Spielekonsole Xbox gibt es das auf .NET aufsetzende XNA-Framework. In Windows Phone 7 arbeiten Silverlight und XNA auf Basis der .NET Compact Framework-Version 3.7.

Seit .NET 2.0 gibt es die Unterstützung für 64-Bit-Systeme (x64, IA64) sowie für den Emulator WOW64, mit dem 32-Bit-Anwendungen auf 64-Bit-Systemen laufen können.



## Silverlight

Silverlight ist eine reduzierte Version des Microsoft .NET Framework mit der man sowohl Plug-In-basierte Browser-Anwendungen als auch eigenständige Desktop-Anwendungen schreiben kann.

Silverlight ist ein Ableger von WPF, also auch XAML-basiert. Der frühere Name war WPF/E (Windows Presentation Foundation Everywhere). Ursprünglich war Silverlight ein reines Browser-Plug-In als Konkurrenz zu Macromedia Flash. Silverlight kann jedoch seit Version 3.0 auch außerhalb des Browsers laufen (*Out-Of-Browser*, kurz OOB). Seit Version 4.0 ist sogar ein Ausbrechen aus der Sandbox mit fast vollständigem Zugang zu lokalen Ressourcen möglich (*Trusted Application*). Silverlight wird damit immer mehr zu einer Konkurrenz der großen Mutter .NET.

Allein vom Begriff her muss man schon aufpassen: WPF ist eine reine Oberflächenbibliothek im Rahmen des .NET Framework und klar abgegrenzt von anderen nicht-visuellen Bibliotheken des .NET Framework. Silverlight ist hingegen ein Oberbegriff für alle Teile des Mini-.NET, sowohl für die visuellen als auch die nicht-visuellen.

Die erste Version von Silverlight, die am 04.09.2007 erschienen ist, bot zunächst XAML als Oberflächenbeschreibungssprache sowie die Programmierbarkeit mit JavaScript. Die zweite Version ist jedoch ein Mini-.NET Framework, das nicht nur XAML, sondern das zahlreiche .NET-Bibliotheken (z.B. Netzwerkprogrammierung mit System.Net, Webservices mit WCF/System.ServiceModel, Abfragen mit LINQ, XML-Verarbeitung mit System.Xml) unterstützt und die Programmierbarkeit mit C#, Visual Basic, Managed JScript, IronRuby und IronPython bietet.

Microsoft bietet Silverlight für die Betriebssysteme Windows Vista, XP, 2000, 2003 und 2008 sowie Apple OS X Tiger und Leopard an. Als Browser werden dabei unterstützt: Internet Explorer (ab 6.0), Firefox (ab 1.5) und Safari (ab 2.0). Im Rahmen von Mono (siehe nächste Abschnitte) gibt es mit *Moonlight* auch eine Unterstützung für Unix/Linux, die Microsoft sogar ausdrücklich fördert [MIC01]. Microsoft spricht daher von Silverlight auch als einem *Cross-Plattform .NET*.

Silverlight verwendet auch XAML zur Oberflächenbeschreibung, aber das Silverlight-XAML ist nicht 100% kompatibel zum WPF-XAML. Zum einen sind – verständlicherweise – nicht alle XAML-Funktionen aus WPF in Silverlight verfügbar. Auf der anderen Seite gibt es aber auch einige syntaktische Eigenarten in Silverlight, die es in WPF noch nicht gibt. Dieses Kompatibilitätsproblem will Microsoft aber in der Zukunft beheben (vgl. Ankündigung [SG03]).

Die Architektur von Silverlight ist auf die Verwendung von Webservices auf einem Application Server ausgelegt (3-Tier-Architektur). Es gibt in Silverlight kein ADO.NET und kein ADO.NET Entity Framework. Silverlight muss sich Daten immer von einem Webservice beschaffen, auch in der OOB-Variante. Dieser Webservice, der dann im einfachsten Fall mit .NET implementiert ist, kann natürlich serverseitig alle Fähigkeiten von .NET nutzen. Allenfalls eine *Trusted Application* könnte über die COM-Objekte der ActiveX Data Objects (ADO) direkt mit einer Datenbank reden.

Aktuell ist Silverlight-Version 4.0, die am 16.4.2010, also sehr kurz nach .NET 4.0, erschienen ist.

Eine weitere Diskussion oder Beschreibung von Silverlight ist in diesem Buch leider aus Platzgründen nicht möglich.

## ECMA SSCLI (Rotor)

Die ECMA stellt seit Mai 2002 drei Referenzimplementierungen der CLI für Windows XP, FreeBSD 4.5 und Mac OS X bereit (die Quellcodes und Dokumente befinden sich nicht auf dem Webserver der ECMA, sondern auf den Homepages der beteiligten Unternehmen, siehe [MSDN23]). Diese Referenzimplementierung wird als *Shared Source CLI (SSCLI)* oder kurz mit dem Codenamen *Rotor* bezeichnet. Wie den Listings von Rotor zu entnehmen ist, stammt der Code von Microsoft. Entfernt sind gegenüber dem .NET Framework lediglich die nicht standardisierten Klassen und Klassenmitglieder. Rotor darf nur für Studienzwecke, nicht aber kommerziell oder für eigene Produkte genutzt werden.

---

**TIPP** Der Rotor-Quellcode stammt zum größten Teil von Microsoft selbst (es handelt sich um eine reduzierte Version des .NET Framework-Quellcodes) und ist eine gute Chance, hinter die Kulissen von .NET zu schauen.

---

**HINWEIS** Die Referenzimplementierung der ECMA, Shared Source CLI (SSCLI), ist im März 2006 in der Version 2.0 erschienen, die dem .NET Framework 2.0 entspricht. Bestrebungen, die Klassen aus .NET 3.0/3.5/4.0 zu standardisieren, gibt es bislang nicht.

---

## Novell Mono

Mit dem Produkt Mono [MONO01] der Firma Novell existiert inzwischen eine ernstzunehmende .NET-Implementierung für Unix, Linux und Mac OS. Mono ist ein Open Source-Projekt und wurde im Jahre 2001 von Miguel de Icaza ins Leben gerufen, der in der Linux-Welt bekannt ist durch den Gnome Desktop, einen Konkurrenten von KDE.

In offiziellen Verlautbarungen bezeichnet die Firma Novell ihr Produkt *Mono* als »eine .NET Implementierung basierend auf den ECMA-Standards für C# und die Common Language Infrastructure«. Tatsächlich geht Mono aber weit über den lückenhaften Standard hinaus: Neben den standardisierten Technologien unterstützt es zusätzlich die Datenzugriffstechnologie ADO.NET, die Webanwendungstechnologie ASP.NET Webforms und auch Windows Forms. Mono bietet zudem weitere Datenbanktreiber und Programmierschnittstellen für spezielle Anwendungsgebiete aus der Linux-Welt.

Mono ist eine komplette Neuentwicklung. Der Quellcode der Shared Source CLI durfte aufgrund der Lizenzbedingungen nicht für die Entwicklung von Mono verwendet werden.

Lange Zeit stand die Frage im Raum, ob Microsoft irgendwann gegen Mono mit patentrechtlichen Argumenten einschreiten wird. Durch die Zusammenarbeitsvereinbarung von Microsoft und Novell aus dem Jahre 2006 (siehe [NOV01]) ist diese Gefahr erstmal gebannt.

---

**HINWEIS** Mono ist insgesamt im Markt noch eine Randerscheinung. Entwickler, die plattformunabhängig arbeiten wollen, setzen eher auf Java.

---

Die zum Redaktionsschluss aktuelle *Mono*-Version 2.8 (6.10.2010) entspricht im Wesentlichen .NET 4.0. Folgende Teile von .NET sind aber im Mono gar nicht verfügbar [MONO02]:

- Windows Presentation Foundation (WPF)
- ADO.NET Entity Framework
- WCF Data Services (Serverseite)

- Windows Workflow
- System.Management
- .NET Enterprise Services

---

**ACHTUNG** Darüber hinaus gibt in vielen Namensräumen Unternamensräume, einzelne Klassen bzw. Klassenmitglieder, die nicht verfügbar sind, z.B. Unternamensraum `System.Net.PeerToPeer`, Klasse `System.Diagnostics.DebuggableAttribute`. Eine detaillierte Liste findet man unter [MONO02]. Das bedeutet, dass sich viele .NET-Anwendungen nicht ohne Nacharbeiten auf Mono neukompilieren lassen.

---

Es gibt Mono-Distributionen für folgende Betriebssystemplattformen:

- Windows
- Mac OS X
- openSUSE
- SUSE Linux Enterprise
- RHEL/CentOS
- Solaris
- Debian
- Ubuntu

---

**TIPP** Auf [MONO06] kann man auch komplette virtuelle Systeme mit openSUSE und vorinstalliertem Mono herunterladen.

---

Ableger von Mono sind Moonlight (das Pendant zu Silverlight), MonoTouch (für iPhone, iPod und iPad) sowie MonoDroid (für Android), die Entwicklungsumgebung MonoDevelop sowie die Mono-Werkzeuge für Visual Studio.

---

**HINWEIS** Die weiteren Ausführungen in diesem Buch basieren auf dem Microsoft .NET Framework. Für Unterschiede in Mono sei auf die Mono-Website [MONO05] verwiesen.

---

## Geschichte und Versionen

Das .NET Framework ist entstanden aus Bemühungen von Microsoft, eine einheitliche Laufzeitumgebung für das Component Object Model (COM) zu entwickeln. COM ist das Komponentenmodell, das Microsoft Anfang der 90er Jahre entwickelt hat und das heute wesentlicher Bestandteil des Windows-Betriebssystems und vieler Windows-Anwendungen von Microsoft und anderen Anbietern ist.

Als sich herausstellte, dass *COM Version 3.0* sich sehr weit von den Vorgängerversionen entfernen würde, hat Microsoft einen neuen Namen vergeben. Zunächst wurde das Konzept *Next Generation Windows Service* (NGWS) genannt, seit Juli 2000 verwendet Microsoft den Begriff *.NET Framework*.

---

**HINWEIS** Dass COM der Ausgangspunkt für die Entwicklung von .NET war, merkt man noch an einigen Stellen in den Eingeweiden des .NET Framework, an denen die Begriffe COM und COM+ erscheinen (z.B. auch im Dateinamen des *.NET Framework SDK Logo: complus.gif*).

---

| .NET-Version | Codename/<br>früherer Name  | Installationsgröße           | Erscheinungstermin | Zugehörige Visual<br>Studio-Version | Alternative Visual<br>Studio-Version  |
|--------------|---|------------------------------|--------------------|-------------------------------------|---|
| 1.0          | COM+ v3/Next<br>Generation Windows<br>Service (NGWS)              | 19.7 MB                      | 5.1.2002           | Visual Studio .NET<br>2002          |   |
| 1.1          | Everett   | 23.1 MB                      | 1.4.2003           | Visual Studio .NET<br>2003          |   |
| 2.0          | Whidbey   | 22.4 MB                      | 7.11.2005          | Visual Studio 2005                  | Visual Studio 2008<br>oder 2010   |
| 3.0          | WinFX   | 50.3 MB<br>(64-Bit: 90.1 MB) | 19.11.2006         | keine                               | Visual Studio 2005 +<br>Erweiterungen oder<br>Visual Studio 2008<br>oder 2010 |
| 3.5          | Orcas   | 197.0 MB                     | 19.11.2007         | Visual Studio 2008                  | Visual Studio 2010  |
| 4.0          | <i>Dev10 und Rosario</i><br>(ursprünglich auch<br><i>Hawaii</i> ) | 49.2 MB                      | 12.4.2010          | Visual Studio 2010                  | keine   |

Tabelle 1.2 Bisher erschienene .NET-Versionen

**HINWEIS**

Microsoft verwendet intern und oft auch in öffentlichen Dokumenten (insbesondere in Weblogeinträgen und Folienvorträgen) Codenamen anstelle der Versionsnummer (siehe Tabelle). Bei den Codenamen handelt es sich um Inseln im Pazifischen Ozean, die aus Sicht des US-Festlandes jeweils weiter draußen im Pazifik liegen. Die Ausnahme dabei bildet Everett, bei der es sich um eine Küstenstadt im US-Bundesstaat Washington handelt.

## .NET 1.x

Die Version 1.0 wurde im Januar 2002 freigegeben. Im April 2003 ist die Version 1.1 (Codename Everett) mit kleineren Verbesserungen und Fehlerbehebungen erschienen.

## .NET 2.0 (Whidbey)

Nachdem im Juli 2003 zunächst nur 500 ausgewählte Alpha-Tester Einblick erhielten, hat Microsoft im Oktober 2003 auf der Professional Developers Conference (PDC) eine Alpha-Version des .NET Framework 2.0 und von Visual Studio 2005 an mehr als 7.000 Entwickler verteilt. In dieser Alpha-Version trug das .NET Framework noch die Versionsnummer 1.2 und die Entwicklungsumgebung hieß Visual Studio .NET 8.0. Die erste Beta-Version von Visual Studio 2005 (ohne .NET im Namen) erschien im Juni 2004, die Beta-2-Version im April 2005. Die finale Version kam am 28. Oktober 2005 auf den Microsoft Servern heraus. Die offiziellen Feierlichkeiten zur Marktübergabe fanden am 7. November 2005 statt.

Zahlreiche Neuerungen findet man im .NET Framework 2.0 sowohl in der Laufzeitumgebung Common Language Runtime (CLR) und in den .NET-Programmiersprachen als auch in der Klassenbibliothek und in den zentralen Schichtenkonzepten ADO.NET, ASP.NET und Windows Forms. Viele Neuerungen haben als Zielsetzung, für Standardaufgaben die Menge des Programmcodes zu reduzieren, den der Entwickler selbst verfassen muss. Außerdem ist mit dem Click-Once-Deployment eine neue Technologie zur Verteilung von Anwendungen über Webserver und Netzwerklaufwerke mit Auto-Update-Funktion enthalten.

## .NET 3.0 (WinFX)

Das .NET Framework 3.0 ist am 6. November 2006 erschienen, fast genau ein Jahr nach .NET 2.0. Der kurze Zyklus zwischen .NET 2.0 und 3.0 kam für viele überraschend, denn Microsoft hatte zunächst gar kein neues .NET Framework für das Jahr 2006 angekündigt, sondern eine ergänzende Klassenbibliothek mit dem Namen *Windows Framework (WinFX)*. Dabei war WinFX bei der allerersten Ankündigung auf der Professional Developers Conference 2003 der Name für eine neue .NET-basierte Programmierschnittstelle in Windows Vista (damals noch *Windows Longhorn Client*). WinFX sollte das veraltete Windows 32-API und das heterogene Sammelsurium von nach dem Component Object Model (COM) entwickelten Softwarekomponenten ablösen.

WinFX hat sich dann weiterentwickelt: Zunächst wurde aus dem Vista-API auf Kundendruck eine Klassenbibliothek, die es auch als Erweiterung zu Windows XP und Windows Server 2003 geben sollte. Schließlich gab es in Redmond die Entscheidung, Vista doch nicht mit Hilfe von .NET neu zu entwickeln. Damit war WinFX endgültig losgelöst von Windows Vista.

Dann diskutierte Microsoft mehr oder weniger intern, ob man .NET und WinFX nicht unter dem Namen WinFX zusammenfassen solle. Schließlich ließ Microsoft den Namen WinFX fallen und gemeindete die neuen Klassen in .NET als .NET Framework 3.0 ein. Den Namen .NET Framework 3.0 hat Microsoft am 9. Juni 2006 »enthüllt«.

.NET 3.0 enthält weder eine neue Laufzeitumgebung noch neue Compiler oder eine neue Sprachsyntax. Das .NET Framework 3.0 ist eine echte Erweiterung zum .NET Framework 2.0. In der Version 3.0 gibt es zusätzlich vier große Bibliotheken:

- Windows Presentation Foundation (WPF), Codename *Avalon*
- Windows Communication Foundation (WCF), Codename *Indigo*
- Windows Workflow Foundation (WF) (die Abkürzung ist tatsächlich nur WF, nicht WWF!)
- Windows CardSpace (WCS), früher: Infocard

Das .NET Framework 3.0 umfasst also:

- die Common Language Runtime Version 2.0 (wie im .NET Framework 2.0)
- die .NET-Sprachen der zweiten Generation, also C# 2.0 und Visual Basic 8.0 (wie in .NET Framework 2.0)
- die .NET Framework Class Library Version 3.0 mit den Klassen der .NET Framework Class Library 2.0 und o.g. zusätzlichen Erweiterungen

---

**ACHTUNG** Damit ist jede .NET 2.0-Anwendung kompatibel mit und lauffähig unter .NET 3.0. Korrekterweise müsste man die Anwendung nicht mehr an dem .NET Framework, sondern an der CLR festmachen. Damit ist klar, dass eine CLR 2.0-Anwendung auch unter .NET 3.0 läuft, weil .NET 3.0 auch die CLR 2.0 enthält. Es zeichnet sich ab, dass Microsoft in der Zukunft die Versionsnummer der CLR und der Klassenbibliothek entkoppeln wird.

---

**HINWEIS** Entgegen der ursprünglichen Ankündigungen der starken Verwendung von .NET wurde Windows Vista letztlich ausschließlich mit klassischen Programmiertechniken entwickelt und alle neuen Funktionen des Betriebssystems stehen für Entwickler auch nur auf klassischen Wegen (C-DLLs oder COM) zur Verfügung.

---

## Erweiterungen zu .NET 2.0/3.0

Um den Jahreswechsel 2006/2007 sind noch zwei wichtige Erweiterungen für .NET 2.0 und 3.0 erschienen:

- die Windows PowerShell 1.0
- die AJAX-Erweiterungen 1.0 für ASP.NET 2.0
- Service Pack 1 für .NET 2.0 und .NET 3.0

Die Service Packs 1 zu .NET 2.0 und .NET 3.0 sind parallel mit dem .NET Framework 3.5 erschienen. Das .NET Framework 3.5 setzt voraus, dass vorher die Service Packs für die Vorversionen installiert sind, nimmt dies aber automatisch bei der Installation vor.

Es gilt also die Formel:

$\text{.NET 3.5} = \text{.NET 2.0} + \text{.NET 2.0 SP1} + \text{.NET 3.0} + \text{.NET 3.0 SP1} + \text{Neue Funktionen}$

Da es Änderungen und Ergänzungen in den Service Packs gab, kann es zu Inkompatibilitäten bei .NET-Anwendungen kommen. Dies wird in Kapitel 4 näher besprochen.

## .NET 3.5 (Orcas)

Auf den Tag genau ein Jahr nach .NET 3.0 ist .NET 3.5 erschienen. .NET 3.5 basiert auf einer leicht veränderten Version der CLR 2.0 (CLR 2.0 Service Pack 1). Enthalten in .NET 3.5 sind neue Versionen der Sprachcompiler C# (Version 9.0 für C# 3.0, siehe dazu Kapitel zu C#) und Visual Basic (Version 9.0) sowie der Bibliotheken ASP.NET, Windows Workflow Foundation, Windows Communication Foundation, Windows Presentation Foundation und einiger .NET-Basisbibliotheken. Zentrale neue Bibliothek ist die universelle Abfragesprache *Language Integrated Query (LINQ)*.

Die folgende Grafik veranschaulicht das Verhältnis von .NET 2.0, 3.0 und 3.5. Mit gestricheltem Rand dargestellt sind Erweiterungen, die nicht Teil des ursprünglichen Installationspakets waren und zusätzlich installiert werden müssen.

## Service Pack 1 für .NET 3.5

Im August 2008 ist das Service Pack 1 für .NET 3.5 erschienen, das erhebliche Erweiterungen mit sich brachte:

- Einen weiteren Objektrelationalen Mapper (*ADO.NET Entity Framework*)
- Erweiterungen für ASP.NET in Hinblick auf AJAX (Unterstützung für Browser Vor/Zurück, Skriptverbindungen) sowie neue Funktionen für URL-Routing, zur Unterstützung des Model-View-Controller-Pattern und für die Rapid Application Development (RAD)-Entwicklung datengetriebener Websites (*ASP.NET Dynamic Data*) sowie eine *EntityDataSource* zum Zugriff auf das ADO.NET Entity Framework
- Ein paar neue Steuerelemente für Windows Forms (*Visual Basic Action Pack* – aber trotz des Namens auch für C#!): *PrintForm*, *LineShape*, *OvalShape*, *RectangleShape*, *DataRepeater*

- Microsoft will durch Veränderungen im Just-In-Time-Compiler die Kaltstartzeit für .NET-Anwendungen um bis zu 40% reduziert haben (dies ist eine Angabe von Microsoft, die nicht vom Autor dieses Buchs selbst nachgemessen wurde)
- Das *.NET Framework Client Profile* ist eine abgespeckte Variante des .NET Framework, die auf alle Bibliotheken verzichtet, die (typischerweise) nur in Serveranwendungen verwendet werden
- .NET Framework Setup Bootstrapper: Automatisches Herunterladen benötigter .NET Framework-Versionen beim Start einer .NET-Anwendung (wahlweise .NET Framework oder .NET Framework Client Profile)
- Erweiterung für das Click-Once-Deployment (Unterstützung für Firefox, eigene Installationsoberfläche, Beschränkung durch Gruppenrichtlinien, u. a.)
- Verbesserungen für WCF
- Leistungssteigerung für WCF-HTTP-Dienste (laut Angabe von Microsoft fünf- bis zehnfach!)
- Vereinfachung der Programmierschnittstelle in Bezug auf Serialisierung
- Verbesserungen für WPF
- Leistungssteigerung für WPF (z. B. bei Animationen und beim Blättern in Listen)
- Verbesserungen bei der Datenbindung in WPF (z. B. Null-Werte und Zeichenkettenformatierung)
- Integration von Direct3D in WPF-Anwendungen
- Neues Webbrowser-Steuerelement
- Neue Überblendungseffekte

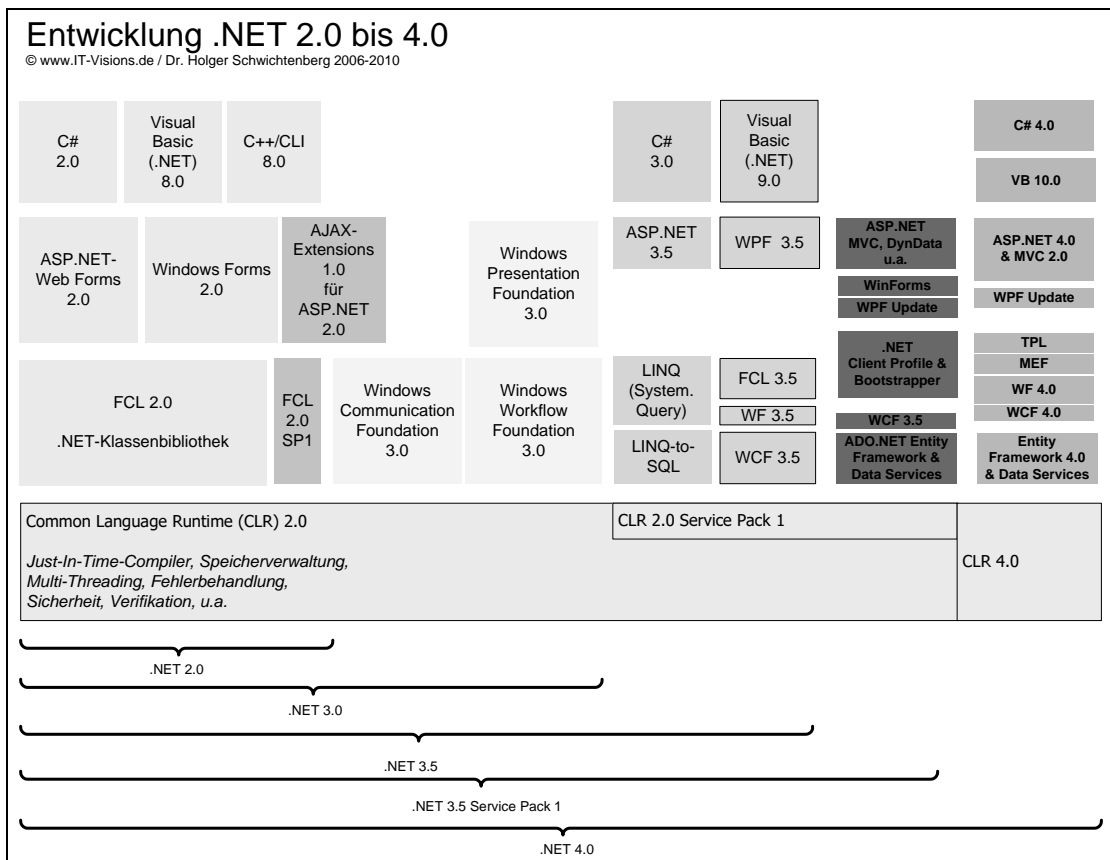
**HINWEIS** Das .NET 3.5 Service Pack 1 sollte besser *Feature Pack* heißen. Noch besser, sollte .NET die Versionsnummer 3.6 oder 3.7 erhalten, denn der Umfang der Neuerungen ist – gerade für ASP.NET und ADO.NET – erheblich.

## .NET 4.0

.NET 4.0 bietet erhebliche Lückenschlüsse und neue Funktionen:

- Erstmals seit .NET 2.0 gibt es wieder eine neue Version der Common Language Runtime (CLR), der Laufzeitumgebung von .NET
- Kleinere Spracherweiterungen in C# 4.0 und Visual Basic 10.0, insbesondere zur Angleichung der Sprache
- ADO.NET Entity Framework 4.0 ist eine erhebliche Verbesserung gegenüber der Vorgängerversion, die in .NET 3.5 SP1 erschienen war
- WCF Data Services 2.0 heißt die erweiterte Version der ADO.NET Data Services aus .NET 3.5 SP1
- ASP.NET 4.0 bietet mehr Neuerungen als ASP.NET 3.5 gegenüber ASP.NET 2.0
- WPF 4.0 enthält nur kleinere Verbesserungen. Wesentlich ist der stark verbesserte Designer in Visual Studio 2010

- Eine vereinfachte Konfiguration und erheblich Erweiterungen (Routing, Discovery) bietet WCF 4.0
- Windows Workflow 4.0 ist eine komplette Neuimplementierung des Workflow-Frameworks in .NET
- Die Erstellung von Add-Ins erleichtert das neue Managed Extensibility Framework (MEF)
- Microsoft vereinfacht die Parallelprogrammierung mit Parallel LINQ und der Task Parallel Library (TPL)
- Darüber hinaus gibt es viele kleinere Ergänzungen in den Basisbibliotheken



**Abbildung 1.8** Entwicklung von .NET 2.0 zu .NET 4.0

## Produkte und Installationspakete

In diesem Kapitel werden die verschiedenen Teilprodukte von Microsoft .NET genannt.



## Notwendige Produkte

Zur Softwareentwicklung mit .NET benötigen Sie auf jeden Fall die in dieser Tabelle genannten Produkte. Theoretisch können Sie das Open Source-Produkt *Sharp Develop* [SHARP01] anstelle von Visual Studio einsetzen. Seit es allerdings kostenfreie Express-Versionen von Visual Studio gibt, verbleiben nur noch wenige Gründe für den Einsatz von Sharp Develop.

| Produktname                                       | Erläuterung  | Anbieter  | Kostenfrei | Bezugsquelle  |
|---|--|-----------|------------|---|
| .NET Framework Redistributable 4.0                | .NET-Laufzeitumgebung (Hinweis: enthalten in Visual Studio 2010)   | Microsoft | Ja         | <a href="http://www.microsoft.com/downloads/details.aspx?FamilyID=0a391abd-25c1-4fc0-919f-b21f31ab88b7&amp;displaylang=de">http://www.microsoft.com/downloads/details.aspx?FamilyID=0a391abd-25c1-4fc0-919f-b21f31ab88b7&amp;displaylang=de</a>       |
| .NET Framework Redistributable 4.0 Client Profile | .NET-Laufzeitumgebung, reduzierte Version nur für Clients (Hinweis: enthalten in .NET Framework Redistributable 4.0) | Microsoft | Ja         | <a href="http://www.microsoft.com/downloads/en/details.aspx?FamilyID=e5ad0459-cbcc-4b4f-97b6-fb17111cf544&amp;displaylang=en">http://www.microsoft.com/downloads/en/details.aspx?FamilyID=e5ad0459-cbcc-4b4f-97b6-fb17111cf544&amp;displaylang=en</a> |
| Silverlight Runtime                               | Ein Mini-.NET für den Browser; Alternative zu Macromedia Flash (Hinweis: enthalten in Visual Studio 2010)            | Microsoft | Ja         | <a href="http://www.silverlight.ent">http://www.silverlight.ent</a>   |
| Windows 7.0 und .NET 4.0 Software Development Kit | Dokumentation, Beispiele und Werkzeuge (Hinweis: enthalten in Visual Studio 2010)                                    | Microsoft | Ja         | <a href="http://www.microsoft.com/downloads/en/details.aspx?FamilyID=6b6c21d2-2006-4afa-9702-529fa782d63b&amp;displaylang=en">http://www.microsoft.com/downloads/en/details.aspx?FamilyID=6b6c21d2-2006-4afa-9702-529fa782d63b&amp;displaylang=en</a> |
| Visual Studio 2010 (VS 2010)                      | Entwicklungsumgebung   | Microsoft | Zum Teil   | Kommerzielle Varianten: Handel<br>Kostenfreie Varianten:<br><a href="http://msdn.microsoft.com/vstudio/express/">http://msdn.microsoft.com/vstudio/express/</a>   |

**Tabelle 1.3** Kernprodukte von .NET

**HINWEISE** Die obige Liste ist aus der Sicht des Softwareentwicklers zusammengestellt. Zur Laufzeit einer .NET-Anwendung wird nur das .NET Framework Redistributable 4.0 und/oder die Silverlight 4.0 Runtime benötigt.

Durch die Installation von Visual Studio 2010 (ab Variante *Professional*) werden alle o. g. Pakete mitinstalliert.

Seit .NET 3.5 Service Pack 1 gibt es bei .NET genau wie bei Java eine Unterscheidung in ein Client- und ein Server-Framework. Das .NET Framework Client Profile ist die reduzierte Client-Version (insbesondere ohne Webentwicklungsklassen). Silverlight ist eine nochmals eingeschränkte Version, die ursprünglich nur für Browser-Plug-In-Anwendungen entwickelt wurde, seit Version 3.0 aber auch außerhalb des Browsers laufen kann. Silverlight ist in diesem Buch nicht näher betrachtet.

## Optionale Produkte

Sowohl kommerzielle Anbieter als auch die Open Source-Welt bieten inzwischen eine reichhaltige Auswahl von Werkzeugen und Softwarekomponenten an, die die von Microsoft bereitgestellte Infrastruktur ergänzen bzw. kostenfreie Lösungen für Werkzeuge offerieren, die Microsoft gegen Entgelt anbietet.

| Produktname                                    | Erläuterung  | Anbieter           | Kostenfrei | Bezugsquelle  |
|--|--|--------------------|------------|---|
| J# 2.0 Redistributable                         | Programmiersprache Java für .NET (wird seit .NET 2.0 leider nicht mehr weiterentwickelt)   | Microsoft          | Ja         | <a href="http://www.microsoft.com/downloads/details.aspx?familyid=F72C74B3-ED0E-4AF8-AE63-2F0E42501BE1&amp;displaylang=en">http://www.microsoft.com/downloads/details.aspx?familyid=F72C74B3-ED0E-4AF8-AE63-2F0E42501BE1&amp;displaylang=en</a> |
| IronPython                                     | Programmiersprache Python für .NET   | Microsoft          | Ja         | <a href="http://www.ironpython.com/">http://www.ironpython.com/</a>   |
| PowerShell                                     | Interaktive .NET-basierte Kommandozeilenschnittstelle  | Microsoft          | Ja         | <a href="http://www.microsoft.com/windowsserver2003/technologies/management/powershell/default.msp">http://www.microsoft.com/windowsserver2003/technologies/management/powershell/default.msp</a>   |
| Team Foundation Server (TFS) 2010              | Entwicklungsserver für die Unterstützung des Anwendungslebenszyklus (Modellierung, Projektmanagement, Aufgaben-/ Fehlerverfolgung und Quellcodeverwaltung, Continuous Integration) | Microsoft          | Nein       | Handel  |
| C# Code Snippets                               | Codefragmente  | Microsoft          | Ja         | <a href="http://msdn.microsoft.com/vstudio/downloads/codesnippets/">http://msdn.microsoft.com/vstudio/downloads/codesnippets/</a>   |
| Refactor! for Visual Basic                     | Refactoring  | Microsoft          | Ja         | <a href="http://msdn.microsoft.com/vbasic/downloads/tools/refactor/">http://msdn.microsoft.com/vbasic/downloads/tools/refactor/</a>   |
| Productivity Power Tools                       | Viele hilfreiche Erweiterungen des Editors   | Microsoft          | Ja         | <a href="http://visualstudiogallery.msdn.microsoft.com/en-us/d0d33361-18e2-46c0-8ff2-4adea1e34fef">http://visualstudiogallery.msdn.microsoft.com/en-us/d0d33361-18e2-46c0-8ff2-4adea1e34fef</a>   |
| Entity Designer Database Generation Power Pack | Erweiterung der Datenbankgenerierungsfunktionen in Visual Studio 2010  | Microsoft          | Ja         | <a href="http://visualstudiogallery.msdn.microsoft.com/en-us/df3541c3-d833-4b65-b942-989e7ec74c87">http://visualstudiogallery.msdn.microsoft.com/en-us/df3541c3-d833-4b65-b942-989e7ec74c87</a>   |
| VS2010 Visualization and Modeling Feature Pack | Erweiterung der UML-Modellierungsfunktionen in Visual Studio 2010  | Microsoft          | Nein       | Nur MSDN-Abonnenten   |
| T4-Editor                                      | Editor für T4-Vorlagen   | Tangible Solutions | Ja         | <a href="http://t4-editor.tangible-engineering.com/T4-Editor-Visual-T4-Editing.html">http://t4-editor.tangible-engineering.com/T4-Editor-Visual-T4-Editing.html</a>   |
| .NET Enterprise Library                        | Klassenbibliothek  | Microsoft          | Ja         | <a href="http://www.codeplex.com/entlib">http://www.codeplex.com/entlib</a>   |
| FxCop  | Quellcodeanalyse   | Microsoft          | Ja         | <a href="http://www.gotdotnet.com/Team/FxCop/">http://www.gotdotnet.com/Team/FxCop/</a>   |
| .NET Reflector                                 | Decompiler   | Lutz Roeder        | Ja         | <a href="http://www.aisto.com/roeder/dotnet/">http://www.aisto.com/roeder/dotnet/</a>   |
| Sandcastle                                     | Dokumentationsgenerierung  | Open Source        | Ja         | <a href="http://sandcastle.codeplex.com/">http://sandcastle.codeplex.com/</a>   |
| NUnit  | Unit Testing   | Open Source        | Ja         | <a href="http://www.nunit.org/">http://www.nunit.org/</a>   |
| NUnitAsp                                       | Unit Testing   | Open Source        | Ja         | <a href="http://nunitasp.sourceforge.net/">http://nunitasp.sourceforge.net/</a>   |
| Subversion                                     | Versionsverwaltung   | Open Source        | Ja         | <a href="http://subversion.tigris.org/">http://subversion.tigris.org/</a>   |
| AnkhSVN  | Versionsverwaltung   | Open Source        | Ja         | <a href="http://ankhsvn.tigris.org/">http://ankhsvn.tigris.org/</a>   |
| TortoiseSVN                                    | Versionsverwaltung   | Open Source        | Ja         | <a href="http://tortoisesvn.tigris.org/">http://tortoisesvn.tigris.org/</a>   |

Tabelle 1.4 Werkzeuge für .NET und Erweiterungen für Visual Studio 2010

**WICHTIG** Die vorstehende Liste ist eine kleine, sehr subjektive Auswahl des Autors. Sie nennt Produkte, die der Autor in seiner Firma selbst verwendet. Insgesamt gibt es mehrere Hundert ergänzende Produkte, die in einer Online-Referenz auf [www.dotnetframework.de/tools.aspx](http://www.dotnetframework.de/tools.aspx) gelistet sind.

## .NET Framework Redistributable

Das .NET Framework Redistributable enthält die Laufzeitumgebung des .NET Framework, d.h. alle notwendigen Bausteine, um .NET-Anwendungen auf einem System ablaufen zu lassen. Das Framework Redistributable muss auf jedem System vorhanden sein, auf dem .NET-Anwendungen ablaufen sollen. Es enthält auch die Kommandozeilen-Compiler für die Programmiersprachen Visual Basic .NET, C# und JScript .NET. Die Kommandozeilen-Compiler ermöglichen die Übersetzung von Quelltexten in den von .NET verwendeten Zwischencode.

Grundsätzlich scheint die Mitgabe von Compilern in einer Laufzeitumgebung überflüssig. Dies ist aber notwendig, da einige Teile der .NET-Klassenbibliothek während der Ausführung von .NET-Anwendungen auf diese Kommandozeilen-Compiler zurückgreifen.

## .NET Framework Client Profile

In der Vergangenheit gab es nur ein .NET Framework Redistributable, das alle Bausteine von .NET sowohl für client- als auch für die serverseitigen Anwendungen enthielt. Der Nachteil war, dass auch auf normalen Desktop-Rechnern serverseitige Komponenten wie ASP.NET installiert wurden, obwohl diese dort gar nicht benötigt werden.

Seit .NET 3.5 Service Pack 1 (d.h. auch in .NET 4.0) gibt es das .NET Framework Client Profile. Das .NET Framework Client Profile ist eine abgespeckte Variante des .NET Framework, die auf alle Bibliotheken verzichtet, die nur in Serveranwendungen verwendet werden. Insbesondere fehlt hier ASP.NET. Visual Studio 2008 SP1 und 2010 stellen eine Kompilierungsoption bereit, die warnt, wenn Bibliotheken verwendet werden, die nicht im .NET Framework Client Profile enthalten sind. Das Kompilat ist aber dann auch lauffähig auf dem vollständigen .NET Framework.

**HINWEIS** Diese Aufspaltung des .NET Framework war längst überfällig. Es ist zu erwarten, dass es weitere Profile für verschiedene Einsatzgebiete geben wird.

## J# Redistributable

Die Programmiersprache J# (gesprochen: »J Sharp«), ein Java-Derivat, ermöglicht die Verwendung von Java als Programmiersprache auf dem .NET Framework. J# besteht aus der Java-Sprachsyntax und einer Implementierung der Java-Klassenbibliothek (*vsjlib*).

**HINWEIS** J# 2.0 ist die aktuelle Version. Es gibt kein J# 3.x. Am 10.1.2007 hat Microsoft angekündigt, dass es im Zuge von Visual Studio 2008 und .NET 3.5 keine neue Version von J# und Visual J# Express geben wird und die Weiterentwicklung von J# eingestellt wird. Die offizielle Unterstützung für J# läuft noch bis 2015 (Visual Studio 2005 + 10 Jahre). J# 2.0 läuft auch auf .NET 4.0, muss dort aber separat heruntergeladen und installiert werden.

## .NET Framework Software Development Kit (.NET SDK) und Windows SDK (WinSDK)

Für die Entwicklung von .NET-Anwendungen zwar nicht notwendig, aber sehr empfehlenswert ist das .NET Framework Software Development Kit (.NET SDK). Das .NET SDK enthält zusätzliche Werkzeuge, die komplette Dokumentation des .NET Framework (einschließlich der Klassenbibliothek) in Form kompilierter HTML-Hilfdateien sowie zahlreiche Beispiele. Das SDK wird im Gegensatz zum Redistributable nur auf den Entwicklersystemen benötigt.

Das Windows SDK (WinSDK) ist eine allgemeine Sammlung von Dokumenten, Beispielen und Werkzeugen zur Windows-Programmierung. Leider liefert Microsoft die SDK-Ergänzungen seit .NET 3.0 nicht als eigenständiges SDK, sondern nur als Teil des Windows SDK.

---

**ACHTUNG** Verwechseln Sie das .NET SDK und das Windows SDK nicht mit dem Visual Studio 2010 Software Development Kit (SDK). Das Visual Studio SDK ist ebenfalls eine Sammlung von Werkzeugen, Dokumentationen und Beispielen, die Sie aber nur benötigen, wenn Sie Erweiterungen (Add-Ins) für Visual Studio entwickeln wollen.

---

## Visual Studio

Lizenzgebühren erhebt Microsoft für die Entwicklungsumgebung *Visual Studio* nur in den Varianten *Standard*, *Premium* oder *Ultimate*. Daneben existieren funktionsreduzierte Versionen mit Namen *Express-Editionen*, die frei erhältlich sind.

Visual Studio (VS) macht die Entwicklung von .NET-Anwendungen durch eine grafische Entwicklungsumgebung wesentlich komfortabler und produktiver im Vergleich zu der Erfassung von Quelltexten mit einem einfachen Texteditor und der Anwendung der Kommandozeilen-Compiler bzw. .NET SDK-Werkzeuge.

---

**HINWEIS** Visual Studio wird von Microsoft schon seit langem als Produktname für die Entwicklungsumgebung verwendet (auch schon vor dem Zeitalter von .NET). Mit der Einführung von .NET erhielten die ersten beiden Versionen den Zusatz .NET: Visual Studio .NET 2002 (alias: Visual Studio 7.0) und Visual Studio .NET 2003 (alias: Visual Studio 7.1). Ab der Version 2005 hat Microsoft wieder auf den Zusatz *.NET* verzichtet. Die aktuelle Version ist Visual Studio 2010.

---

## Visual Studio Ultimate und Team Foundation Server (TFS) (früher: Visual Studio Team System (VSTS))

Das Application Lifecycle Management (ALM) hatte Microsoft viele Jahre nur stiefmütterlich behandelt. Die Quellcodeverwaltung Visual SourceSafe (VSS) war nicht nur funktionsarm, sondern auch für defekte Datenbanken bekannt, die viel Wartungsaufwand erforderten. Für die Modellierung bot Microsoft mit Visio ebenfalls keine adäquate Lösung. Intern verwendete Microsoft schon länger einige selbstentwickelte Werkzeuge, die dann im Jahr 2005 unter dem Oberbegriff Visual Studio Team System (VSTS) für die Öffentlichkeit als eine integrierte ALM-Lösung erschienen. VSTS hatte Microsoft aber nicht zuletzt auch für die eigenen Entwicklungsteams geschaffen. Zu VSTS gehörte einerseits eine erweiterte Version von Visual Studio als Client-Anwendung, zum anderen der Team Foundation Server (TFS) als zentraler Server zur Verwaltung und Steuerung von Softwareprojekten. Damit wurde Visual Studio zu einer echten Client-Server-Lösung.

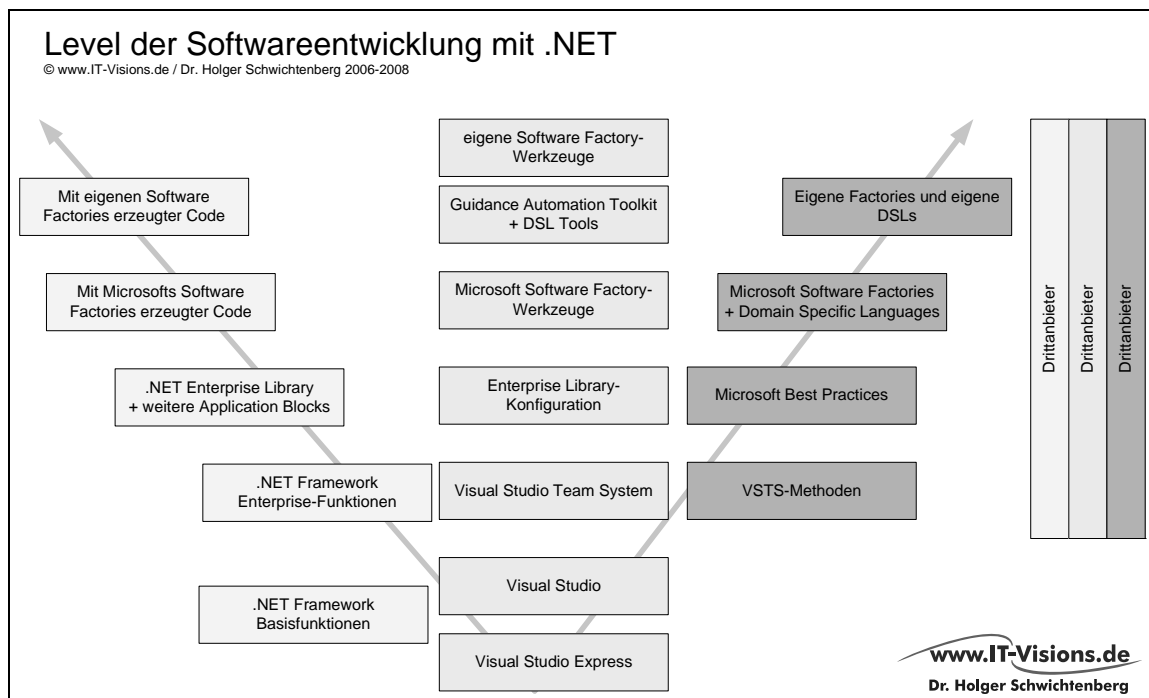
Die Vergangenheitsform ist für den Begriff *VSTS* gerechtfertigt, da Microsoft diesen Begriff mit Visual Studio 2010 nun wieder begräbt. Es gibt weiterhin Visual Studio und den Team Foundation Server, aber keine eigenständige Produktbezeichnung für die ALM-Produktserie mehr. Anstelle der Visual Studio *Team Suite*, die bisher die größte und teuerste Version war, tritt nun *Visual Studio Ultimate*. Einige ALM-Funktionen sind insgesamt noch nicht so weit, wie man es von den Platzhirschen wie Micro Focus (früher Borland), IBM (früher Rational), Perforce und MKS kennt. Beispielsweise ist IBM Rational DOORS sehr viel strukturierter und mächtiger in Bezug auf das Anforderungsmanagement als die Tätigkeitsverwaltung im TFS. Micro Focus bietet mit Borland Together ein Werkzeug für die modellgetriebene Entwicklung, das es bisher in dieser Form bei Microsoft gar nicht gibt. Perforce bietet in seiner Versionsverwaltung einen Verteilungsmechanismus an, um mehrere Versionsverwaltungsserver zu verbinden. In TFS ist ein solch lose gekoppeltes Kooperationsmodell bisher nicht vorhanden. Die anderen Anbieter decken zudem zahlreiche Plattformen ab, nicht nur als Client, sondern auch als Server. Aber die Microsoft-Lösung ist kostengünstiger. Gartner hat dazu im März 2009 einen Vergleich [GART01] veröffentlicht.

## Weitere Werkzeuge

Neben Visual Studio und Team Foundation Server bietet Microsoft inzwischen weitere Werkzeuge für Entwickler mit höheren Ansprüchen und/oder komplexeren Projekten an. Dies sind insbesondere die .NET Enterprise Library und die Software Factories.

*Software Factories* sind ein von Microsoft propagierter Ansatz, mit dem auf Basis der Wiederverwendung von komponentenbasierten Frameworks der Entwicklungsprozess von Anwendungen beschleunigt werden soll. Software Factories sollen auf domänenspezifischen Sprachen (Domain Specific Language – DSL) aufsetzen, wodurch Code-Generierung und andere Formen der Automatisierung in der Softwareentwicklung möglich werden. Die bisher von Microsoft vorgestellten Software Factories erinnern aber mehr an den klassischen Assistenten zur Codegenerierung denn an einen wirklich innovativen Ansatz. Mithilfe des sogenannten *Guidance Automation Toolkit* und der *Domain Specific Language Tools (DSL Tools)* können fortgeschrittene Entwickler eigene Software Factories entwickeln und für andere Entwickler bereitstellen.

Die *.NET Enterprise Library* ist eine im Quellcode verbreitete Klassenbibliothek, die einige Funktionen konkretisiert, die in der .NET-Klassenbibliothek sehr allgemein gehalten sind. Entstanden ist die .NET Enterprise Library innerhalb der Pattern & Practices-Gruppe bei Microsoft, die die Aufgabe hat, .NET-Entwicklern geeignete Handlungsrichtlinien für den Einsatz von .NET-Technologien zu vermitteln. Diese Gruppe hat mehrere so genannter Anwendungsblöcke (Application Blocks) erstellt, die die Anwendung von bestimmten Teilen der .NET-Klassenbibliothek vereinfachen und für das Umfeld der Enterprise-Anwendungsentwicklung nutzbar machen. Die Anwendungsblöcke zeigen Lösungen für typische Entwicklungsaufgaben in großen, mehrschichtigen, verteilten Anwendungen auf. Ab Version 3.0 der .NET Enterprise Library erhält man zusätzlich auch von Microsoft vorkompilierte und digital signierte Kompilate.



**Abbildung 1.9** Softwareentwicklung mit .NET kann man auf verschiedenen Ebene betreiben: Auf der untersten Ebene mit dem kostenlosen Visual Studio Express oder auf höchster Ebene mit Visual Studio Team System unter Einsatz von Software Factories

## Entscheidung zwischen der deutschen und der englischen Version

Viele Entwickler stellen dem Autor dieses Buchs in Schulungen oder bei Vorträgen die Frage, ob man die deutsche oder englische Version von Visual Studio einsetzen solle.

Dazu ist zunächst Folgendes festzuhalten:

- Auf die entstehende Anwendung hat die Sprachwahl der Entwicklungsumgebung keinen Einfluss. Sie können grundsätzlich die Anzeigetexte von Steuerelementen frei wählen. In welcher Sprache Fehlermeldungen oder andere vordefinierte Texte angezeigt werden, hängt von den installierten Sprachpaketen des .NET Framework (.NET Framework Language Packs, siehe [MS04]) auf dem Zielsystem ab. Durch Installation entsprechender Sprachpakete (sogenannter *Language Packs*) auf den Produkktivsystemen verfügen Programme, selbst wenn sie mit einer englischen Visual Studio-Version entwickelt wurden, später über deutsche Dialogfelder (wie sie bei Fehlermeldungen oder beim Drucken zu sehen sind).
- Die deutsche Version hat grundsätzlich nicht mehr Programmfehler als die englische Version. Im Einzelfall kann dies zwar nicht ausgeschlossen werden, es ist aber nicht auffallend häufig.

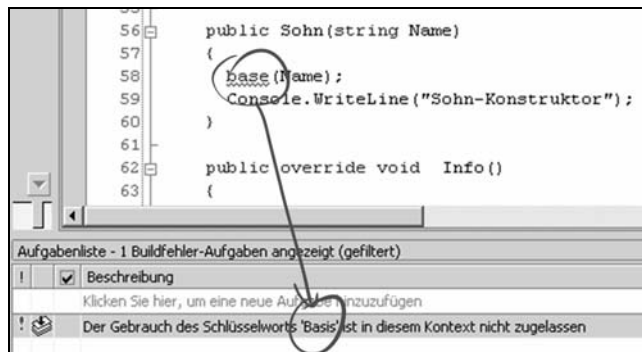
Dennoch entwickelt der Autor dieses Buchs seine .NET-Anwendungen grundsätzlich mit einer englischen Version. Nur zur Erstellung einer Voraufgabe dieses Buchs hat er auf ausdrücklichen Wunsch des Verlages eine deutsche Version eingesetzt (deshalb zeigen einige Bildschirmabbildungen die deutsche Version). Die deutsche Version hat folgende Nachteile:

- Die Übersetzung ist – trotz aller Bemühungen von Microsoft, diese zu verbessern – auch in der Visual Studio 2010-Version immer noch sehr schlecht. Unten sehen Sie einige Beispiele für nicht nur unglückliche, sondern sinnentstellende Übersetzungen.
- Auch in der deutschen Version sind viele Anzeigen (Klassennamen, Attributnamen, Werteoptionen) in Englisch. In der Hilfe sind aber viele dieser Begriffe auch ins Deutsche übersetzt worden. Damit ist die Hilfe nur noch sehr schwer zu verstehen.
- Da es auf der Welt mehr englischsprachige als deutschsprachige Entwickler gibt, wird man bei der Suche nach Hilfen und Tipps im World Wide Web besser fündig, wenn man den englischen Fehlertext in eine Suchmaschine eingibt
- Einige Visual Studio-Erweiterungen sind derzeit nur in englischer Sprache verfügbar. Diese Werkzeuge können Sie dann entweder gar nicht installieren oder aber Sie erhalten dann eine Mischung aus deutscher und englischer Benutzeroberfläche.

### Beispiele für Übersetzungsfehler

Wie die nachstehenden Bildschirmabbildungen beweisen, gibt es auch in der deutschen Version von Visual Studio immer wieder Übersetzungsfehler.

In der ersten Abbildung hat der Compiler natürlich Recht, dass *base* hier verboten ist. Aber das Schlüsselwort *base* heißt auch genau so in der deutschen Version. *Basis* gibt es hingegen nicht.



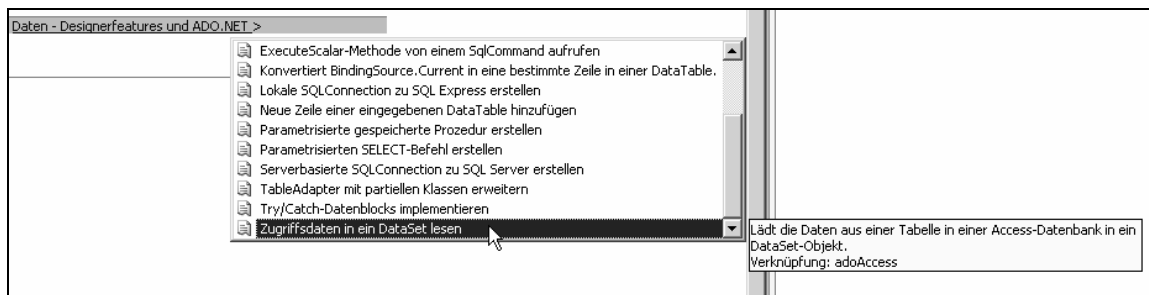
**Abbildung 1.10** Übersetzungsfehler in der deutschen Version von Visual Studio

Der in der folgenden Abbildung erwähnte *eigene Namespace* heißt im englischen Original *My Namespace*, wobei *My* ein Schlüsselwort ist und damit nicht hätte übersetzt werden dürfen, während *Namespace* sich für eine Übersetzung angeboten hätte.



**Abbildung 1.11** Übersetzungsfehler in der deutschen Version von Visual Studio

Sehr lustig ist der Übersetzungsfehler in einem Code Snippet. Was ist wohl mit *Zugriffsdaten in ein DataSet lesen* gemeint? In der englischen Version heißt es *Read Access Data into a DataSet*. Hier war also eine Microsoft Access-Datenbank gemeint, nicht *Zugriffsdaten*. Wären Sie darauf gekommen?



**Abbildung 1.12** Übersetzungsfehler in der deutschen Version von Visual Studio

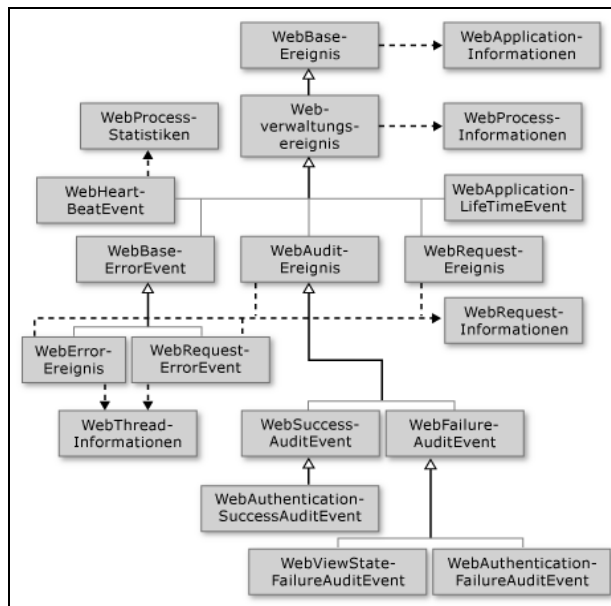
Das *Schießen des Dialogfelds beenden* heißt es in der deutschen Dokumentation. Selbst wenn man sich das fehlende »l« hinzudenkt, bleibt der Satz schwer verständlich. Im Englischen heißt es *stopp the closing*. Eine verständliche deutsche Übersetzung wäre also gewesen: *verhindern, dass sich das Dialogfeld schließt*.

**Hinweis** Mit Hilfe des `Form.Closing`-Ereignisses des Formulars können Sie auch das Schießen des Dialogfelds beenden.

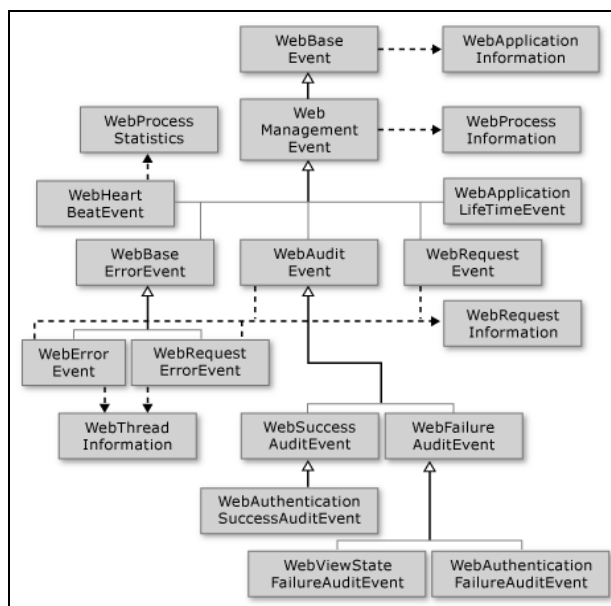
**Abbildung 1.13** Übersetzungsfehler in der deutschen MSDN-Library zu Visual Studio

Ein weiteres ganz typisches Beispiel illustriert die Gegenüberstellung einer Abbildung aus der deutschen und der englischen Dokumentation. In der englischen Dokumentation zeigt das Diagramm eine Hierarchie von Klassennamen. In der deutschen Version wurden diese Klassennamen zum Teil übersetzt (*Webverwaltungsereignis* statt *WebManagementEvent* und *WebBase-Ereignis* statt *WebBaseEvent*). Das ist natürlich ein schwerer Fauxpas, weil ein Entwickler, der diese Abbildung sieht, nur schwerlich jemals die passenden Klassen finden wird.





**Abbildung 1.14** [ms-help://MS.VSCC.v80/MS.MSDN.v80/MS.VisualStudio.v80.de/dv\\_aspnetcon/html/e4930af7-2528-443c-b5fe-dc49b3c047ba.htm](ms-help://MS.VSCC.v80/MS.MSDN.v80/MS.VisualStudio.v80.de/dv_aspnetcon/html/e4930af7-2528-443c-b5fe-dc49b3c047ba.htm)



**Abbildung 1.15** [ms-help://MS.VSCC.v80/MS.MSDN.v80/MS.VisualStudio.v80.en/dv\\_aspnetcon/html/e4930af7-2528-443c-b5fe-dc49b3c047ba.htm](ms-help://MS.VSCC.v80/MS.MSDN.v80/MS.VisualStudio.v80.en/dv_aspnetcon/html/e4930af7-2528-443c-b5fe-dc49b3c047ba.htm)

## Fazit

Diese vorangegangene Aufstellung sollte primär der Erheiterung des Lesers dienen. Die Frage, ob man eine deutsche oder eine englische Version von Visual Studio einsetzen möchte, ist Geschmackssache des jeweiligen Entwicklers. Ein nicht unerheblicher Teil der Entwickler in Deutschland nutzt eine englische Version.

## Installation

Dieses Kapitel gibt Hinweise zur Installation von .NET und Visual Studio.

**HINWEIS**

Die Installationen von .NET Framework Redistributable und Visual Studio 2010 erfordern Administratorrechte auf dem Zielsystem. Die spätere Nutzung ist auch für normale Benutzer möglich.

## Betriebssysteme, die bereits .NET enthalten

Der Windows Server 2003 war das erste Betriebssystem, bei dem das .NET Framework Redistributable (in der Version 1.1) mit zum Liefer- und Standardinstallationsumfang gehörte. Für alle anderen Windows-Betriebssysteme ab Windows 98 ist das .NET Framework 1.1 als kostenlose Erweiterung verfügbar. Das .NET Framework 2.0 wird zusammen mit Windows Server 2003 Release 2 (R2), Vista und Server 2008 ausgeliefert. Windows Vista enthält auch das .NET Framework 3.0. Windows Server 2008 enthält .NET 2.0 und .NET 3.0 ist eine Option.

Das .NET Framework 2.0 Redistributable kann als Erweiterung auf folgenden Betriebssystemen installiert werden: Windows 2000 (ab Service Pack 4), Windows XP und Windows Server 2003. Eine Installation auf Windows Vista und Windows Server 2008 ist nicht notwendig, da .NET 2.0 eine echte Untermenge von .NET 3.0 ist.

Das .NET Framework Redistributable in den Versionen 3.0 kann also als Erweiterung auf folgenden Betriebssystemen installiert werden: Windows XP und Windows Server 2003. Windows Vista enthält bereits .NET 3.0. Windows Server 2008 enthält automatisch .NET 2.0. .NET 3.0 kann dort als *Feature* (bzw. im Rahmen der Serverrolle *Application Server*) nachinstalliert werden.

Das .NET Framework 3.0 installiert man unter Windows Server 2008 in der Funktion *Add Features*. Zu beachten ist aber, dass die zusammen mit der Sektion *.NET Framework 3.0* genannte Option *WCF Activation* – zur Integration von WCF in den Internet Information Server 7.0 (IIS) bzw. den Windows Process Activation Service (WAS) – erfordert, dass auch der Webserver installiert wird. Windows Server 2008 macht Sie auf diese Abhängigkeiten aufmerksam. Die folgenden Bildschirmabbildungen zeigen den Vorgang.

In Windows Server 2008 R2 ist ebenfalls automatisch .NET 2.0 installiert. Durch *Add Features* kann man .NET 3.5 SP1 (alias 3.5.1) installieren.

Windows 7 enthält .NET 3.5 SP1. Das .NET 4.0 Client Profile bietet Microsoft als optionales Windows Update an.

**ACHTUNG**

Auf Windows Server 2008 Core, dem *Windows ohne Windows*, ist das .NET Framework nicht verfügbar. Auf Windows Server 2008 R2 Core gibt es hingegen eine reduzierte Version des .NET Framework 3.5.

## Installation des .NET Framework 4.0 Redistributable

.NET 4.0 kann als Add-On installiert werden auf Windows XP (ab SP2), Windows Server 2003, Windows Vista und Windows Server 2008 inkl. R2. Das .NET Framework 4.0 kann man installieren durch Installation des .NET Framework Redistributable 4.0 bzw. des zugehörigen Clients Profiles. Bei einer Installation von Visual Studio 2010 wird es automatisch installiert.

Wenn man von einem System ohne .NET ausgeht, werden Sie nach der Installation des Framework Redistributable 4.0 u.a. folgende Veränderungen auf Ihrem System feststellen:

- Es gibt im Windows-Verzeichnis ein Unterverzeichnis *%Assembly%*, in dem der Global Assembly Cache (GAC) mit global zugänglichen Softwarekomponenten residiert
- Im der Softwareverwaltung erscheint *.NET Framework 4 Client Profile* bzw. *.NET Framework 4 Extended* (sofern das komplette .NET Framework Redistribuable installiert wurde)
- Es gibt im Windows-Verzeichnis ein Unterverzeichnis */Framework* (auf 64-Bit-Systemen zusätzlich auch ein *\Framework64*) und darunter *.NET 4.0* mit DLLs, Kommandozeilenwerkzeugen, XML-Dateien, SQL-Skripten u.a.
- In der Konfiguration des Internet Information Server (IIS) – soweit bereits vorher installiert – werden ein ISAPI-Filter und eine ISAPI-Extension hinzugefügt (sofern der IIS vor der Installation des .NET Framework installiert war)
- Es gibt ein neues Benutzerkonto *ASPNET* und eine Benutzergruppe *Debuggerbenutzer* bzw. *Debugger Users*

**HINWEIS** Auf 64-Bit-Betriebssystemen gibt es zwei Verzeichnisse *Framework* und *Framework64*. *Framework* ist die 32-Bit-Version, *Framework64* entsprechend die 64-Bit-Version.

Für die Installation des .NET Framework 4.0 ist manchmal, aber nicht immer, ein Neustart des Systems nach Abschluss notwendig.

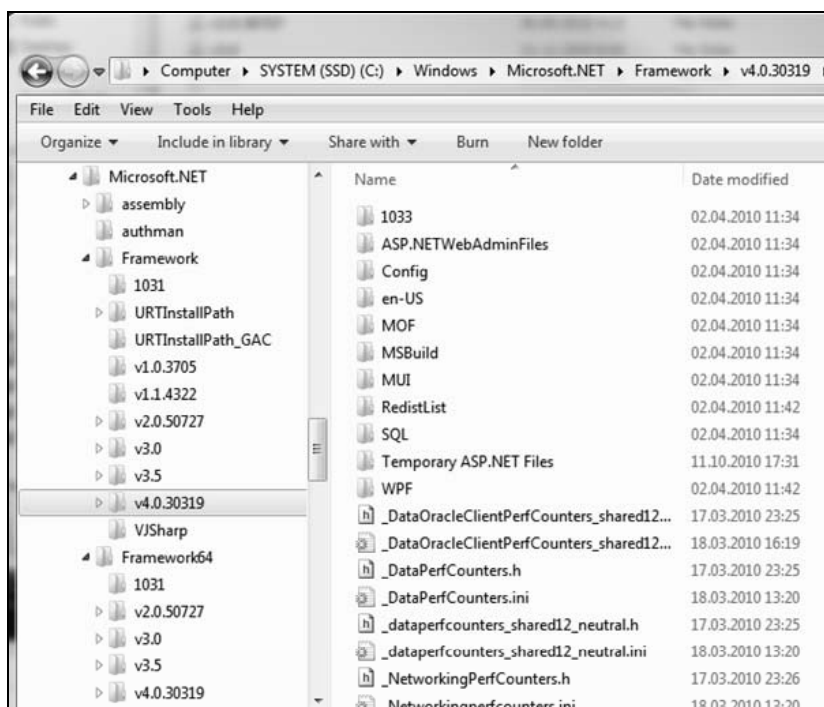
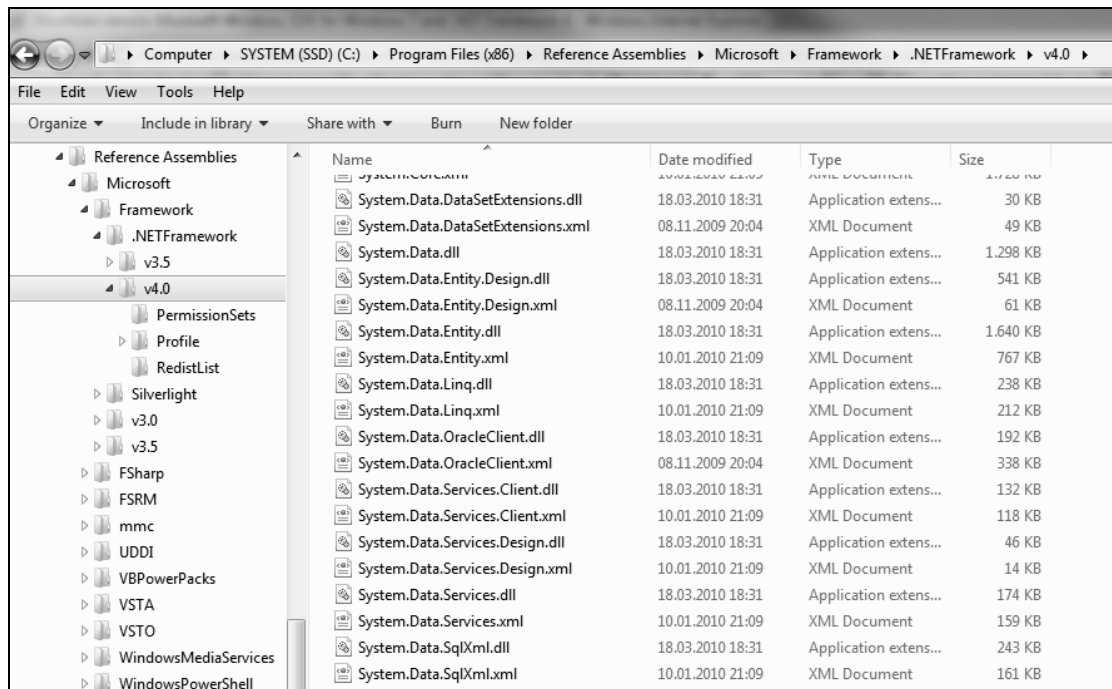


Abbildung 1.16 Ein Blick in das *Microsoft .NET*-Verzeichnis

Abbildung 1.17 Ein Blick in das *Reference Assemblies*-Verzeichnis

## .NET Software Development Kit (SDK)

Das .NET SDK enthält eine Reihe von Zusatzwerkzeugen (z.B. *ildasm.exe* zur Dekompilierung, *gacutil.exe* zur Aufnahme einer Assembly in den Global Assembly Cache, *sn.exe* zum Signieren einer Assembly, *SvcUtil.exe* zum Generieren eines WCF-Dienstproxies). Durch die Installation des SDK wird auch das Verzeichnis `%Programme%\Reference Assemblies\Microsoft\Framework\.NETFramework\v4.0` angelegt, das eine Kopie aller Bibliotheken (Assemblies) von .NET 4.0 enthält für den vereinfachten Zugriff aus Entwicklungsumgebungen heraus.

**HINWEIS** In den ersten Versionen gab es das .NET Software Development Kit (SDK) als eigenständigen Download. Im Zuge von Windows Vista wurde es in das umfassende Windows SDK integriert. Die aktuelle Version heißt Microsoft Windows SDK for Windows 7 and .NET Framework 4 [<http://www.microsoft.com/downloads/en/details.aspx?FamilyID=6b6c21d2-2006-4afa-9702-529fa782d63b&displaylang=en>].

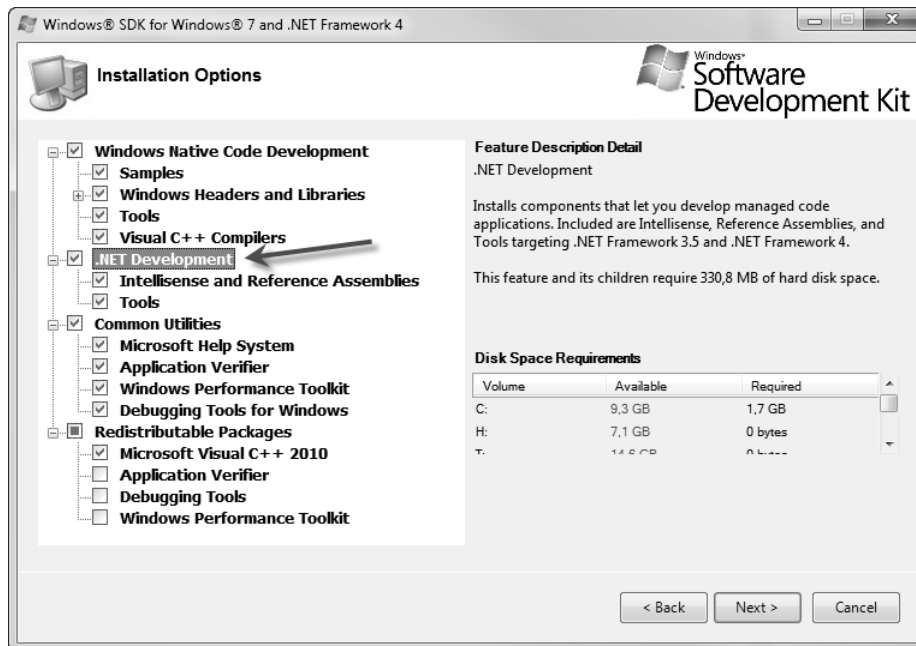


Abbildung 1.18 .NET-Bestandteile des Windows SDK

## Installation von Visual Studio 2010

Die Entwicklungsumgebung Visual Studio 2010 (gilt für alle Varianten) kann nur auf XP (ab SP3), 2003, Vista (jeweils ab SP 2) und 2008 (mit SP2) sowie Windows 7 und 2008 R2 installiert werden. Die frühere Version Visual Studio 2005 konnte auch noch auf Windows 2000 (mit SP4) installiert werden.

**HINWEIS** Eine 64-Bit-Variante von Visual Studio gibt es auch im Zeitalter von Visual Studio 2010 leider nicht. Das 32-Bit Visual Studio kann aber auf x64-Systemen (nicht aber auf IA64-Systemen) installiert werden. In dieser x64-Konstellation existieren jedoch Einschränkungen beim Debugging. Diese werden im Kapitel 5 »Visual Studio 2010« thematisiert, um hier Platz zu sparen.

Rico Mariani, Softwarearchitekt bei Microsoft, begründet die Entscheidung gegen eine baldige 64-Bit-Migration in einem Weblog [<http://blogs.msdn.com/b/ricom/archive/2009/06/10/visual-studio-why-is-there-no-64-bit-version.aspx>]. Leider ist und bleibt des Debugging mit Visual Studio auf 64-Bit-Systemen eingeschränkt. Nicht möglich im 64-Bit-Modus sind Edit-Debug-Continue und IntelliTrace. Um die vollen Funktionen zu nutzen, muss man die Anwendung zum Debugging in den 32-Bit-Modus zwingen, indem man als Zielpattform beim Kompilieren x86 im Auswahldialogfeld *Solution Plattform* wählt.

Bei der Installation von Visual Studio 2010 werden automatisch auch .NET Framework Redistributable 4.0 und .NET Software Development Kit (SDK) installiert. Abhängig von den gewählten Installationsoptionen wird dann auch installiert

- die kostenlose Express Edition der Datenbank Microsoft SQL Server 2008,
- das .NET Compact Framework zum Entwickeln mobiler Anwendungen.

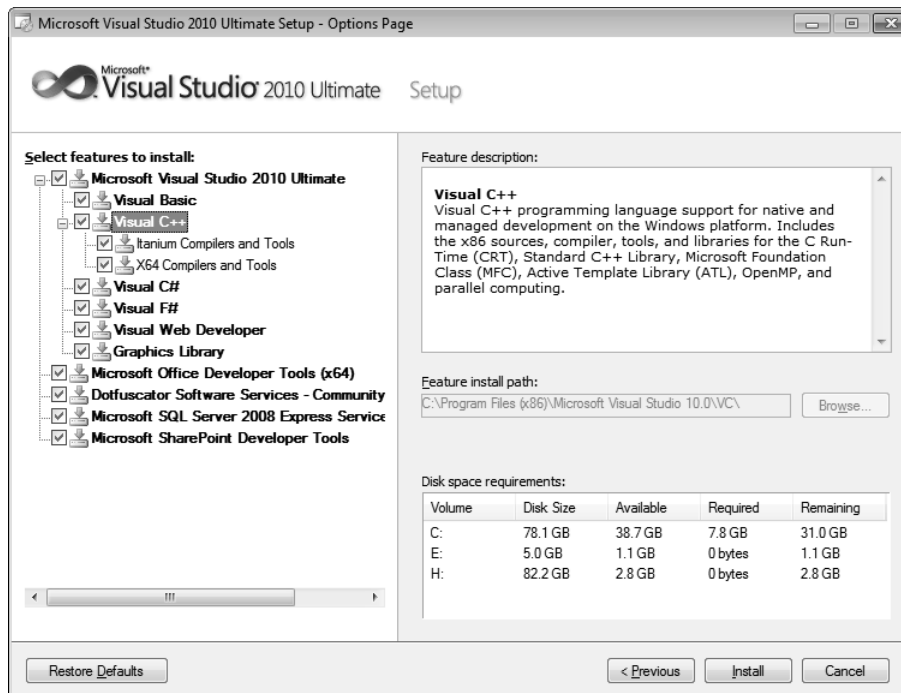


Abbildung 1.19 Installationsoptionen für Visual Studio 2010 (Variante Ultimate)

## Service Pack

Zum Redaktionsschluss dieses Buchs sind Informationen über das erste Service Pack für Visual Studio 2010 noch nicht öffentlich verfügbar.

## Umgebungseinstellungen

Beim ersten Start erwartet Visual Studio 2010 von dem Entwickler die Auswahl einer *Standardeinstellung für die Umgebung*. Die Auswahl *Allgemeine Entwicklungseinstellungen* ist die beste Option für Entwickler, die mehrere Programmiersprachen einsetzen. In jeder Einstellung sind alle Funktionen verfügbar. Menüaufteilung und Tastaturkürzel sind jedoch zum Teil verschieden. Daher ist die Wahl hier Geschmackssache.