Universität Siegen
Diplomstudiengang
Angewandte Informatik
Anwendungsfach Medienwissenschaften

Seminar Softwarearchitekturen
(Dipl. - Inform. Benedikt Meurer)

Sommersemester 2011

Martin Schrage
martin.schrage@googlemail.com

Inhaltsverzeichnis

1. Entstehung von .NET

Das .NET Framework entstand, aus der Anfang der 90er Jahre entwickelten Plattformtechnik Component Object Model(COM). Das COM - Framework wurde vor allem zur Interprozesskommunikation und zur dynamischen Objekterzeugung verwendet und ist auch heute noch ein wesentlicher Bestandteil aller Windows Betriebssysteme und deren Anwendungen. Als sich herausstellte, dass die COM Version 3.0 sich stark von den vorherigen Versionen unterscheiden würde, nannte Microsoft das Framework Next Generation Windows Service(NGWS). Im Juli 2001 wurde der Name .NET Framework eingeführt.

Der zweite Grund für die Einführung von .NET war, dass Microsoft Java von Sun in eine eigene Sprache J++ adaptiert hat, woraus ein längerer Rechtsstreit entstand. Denn J++ war vollständig zu Java kompatibel aber Microsoft erweiterte den Syntax nach eigenen Bedürfnissen. 2004 gab Microsoft die Entwicklung von J++ auf, führte aber J#(J Sharp) in das .NET Framework ein. J# soll wohl Java Entwicklern den Einstieg in .NET und die Übernahme von existierenden Quellcode erleichtern und soll sie zum Verwenden der Sprache C# animieren. Seit 2007 ist auch die Weiterentwicklung von J# eingestellt und die offizielle Unterstützung endet 2015.

Hinzu kommt noch die Inkompatibilität der häufigsten unter Windows benutzten Sprachen, Visual C++, Visual Basic und J++. Dies äußerte sich dadurch, dass die Datentypen der drei Sprachen nicht binärkompatibel zu einander waren und bei der Speicherverwaltung unterschiedliche Konzepte verfolgt wurden. [1]

2008 veröffentlichte Microsoft den kompletten Quellcode des .NET Framework unter der sehr strengen Microsoft Reference Source License, die es nicht erlaubt den Quellcode zu verändern oder in andere Projekte zu integrieren. Dieses Lizenzmodel ermöglicht nur die Einsichtnahme in den Code, um bei Fehlern schnell geeignete Gegenmaßnahmen ergreifen zu können. [2]

2. Common Language Infrastructur

Einige Teile des .NET Framework sind von Microsoft 2001 unter dem Namen Common Language Infrastructur(CLI) bei der European Computer Manufacturers Association als Standard ECMA - 335[3] standardisiert worden, nach kleinen Anpassungen 2002 auch von der International Standardisation Organisation (ISO). Hier durch entstanden einige begriffliche Unterschiede, da Microsoft manche Teile anders bezeichnet als die beiden Standardisierungsorganisationen. Der CLI Standard gliedert sich in sechs sogenannte Partitionen, die sich wie folgt zusammensetzen.

[1] (Schwichtenberg, Microsoft - .NET - 4.0 - Crashkurs, 2011) S.59ff.

[2] (Microsoft Reference Source License)

[3] (ECMA - 335: Common Language Infrastructure , 2010)

2.1 Partition I: Konzept und Architektur

Beschreibt die Konzepte und die gesamte Architektur von CLI und besteht aus folgenden Komponenten:

Common Type System

Das Common Type System (CTS) definiert Regeln, wie Datentypen und deren Werte in dem Hauptspeicher abgebildet werden und welche Operationen auf ihnen erlaub sind. Des Weiteren werden die Typhierarchien, Zugriffsrechte, Sichtbarkeit und die Speicherbereinigung von Objekten der Datentypen definiert.

Hierdurch wird die typsichere Kommunikation zwischen unterschiedlichen Programmiersprachen, sowie die reibungslose Interaktion ihrer Objekte erreicht. Die unterschiedlichen CLI fähigen Programmiersprachen können, müssen aber nicht alle dieselben Datentypen implementieren.

Virtual Execution System

Das Virtual Execution System(VES) spezifiziert die Laufzeitumgebung. Es definiert in einer virtuellen Maschine, Methoden um die Regeln des CTS durchzusetzen. Außerdem werden hier das Laden und Ausführen, der Programmablauf und die Fehlerbehandlung spezifiziert.

Common Language Specification

Die Common Language Specification(CLS) spezifiziert eine Teilmenge der Regeln des CTS. Diese müssen von allen CLS unterstützenden Sprache implementiert werden, da die Funktionen und Datentypen die Basisdienste aller CLI Sprachen darstellen.

2.2 Partition II: Metadaten

Definiert den Aufbau und die Struktur der Metadaten sowie ihr Dateiformat. Die Logik und die Semantik werden in Form eines beispielhaften Assemblers erläutert.

2.3 Partition III: CLI Befehlssatz

Definiert den Befehlssatz der Common Intermediate Language(CIL).

2.4 Partition IV: Klassenbibliotheken

Hier wird ein Überblick über alle CLI Klassen und Bibliotheken gegeben. In Form von XML Dokumenten werden alle Klassen, Datentypen und Schnittstellen beschrieben.

2.5 Partition V: Debug

Beschreibt wie Debug - Informationen zwischen CLI Erzeugern und Verbrauchern ausgetauscht werden können und stellt mit dem Debug Interchange Format ein einheitliches Dateiformat bereit.

2.6 Partition VI: Anhänge

Stellt Beispielprogramme, die in CIL Assembly Language geschrieben sind, bereit. Weiter werden Tools zur Bearbeitung von CLI bereitgestellt und es enthält eine genauere Beschreibung der CLI Instruktionen.

3. Microsoft .NET Framework

Das .NET Framework beruht wie Java auf dem Programmierparadigma Write Once Run Anywhere (WORA). Dies besagt, dass ein einmal entwickelter und compelierter Quellcode auf möglichst vielen verschiedenen Betriebssystemen lauffähig sein soll und das, ohne für jede Umgebung einen eigenen Code erzeugen zu müssen. Außerdem ist es möglich, innerhalb eines Projekts eine Vielzahl von .NET kompatiblen Hochsprachen zu nutzen. Das .NET Framework besteht im Wesentlichen aus zwei Bestandteilen. Zum Einen aus der Common Language Runtime(CLR) und zum Andern aus einer Klassenbibliothek.[4]

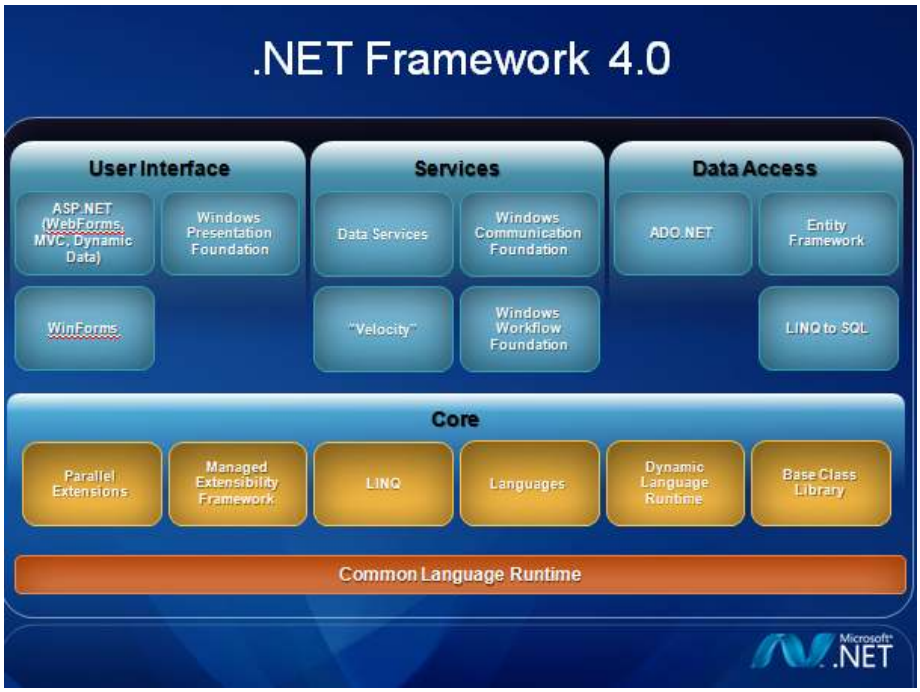

Abbildung 1: Komponenten des .NET Framework[5]

3.1 Common Language Runtime(CLR)

Die CLR[6] ist die Microsoft Implementierung der CLI und damit die Laufzeitumgebung für .NET. So wird das Problem gelöst, dass jede Programmiersprache ihre eigene plattformabhängige

[4] (Microsoft: Konzeptionelle Übersicht über das .NET Framework)

[5] (.NET Framework 4.0)

Laufzeitumgebung benötigt und das diese sehr unterschiedliche Dienste bereitstellen. Die CLR stellt den Just - in - Time Compiler und eine Vielzahl von Basisdiensten bereit, unter anderen folgende:[7]

- eine automatische Speicherverwaltung durch einen Garbage Collector
- ein System zur Ausnahmebehandlung(Exception Handling)
- ein Sicherheitssystem das die Anwender vor Schadsoftware schützt
- durch Application Domains wird die Anwendung von anderen Anwendungen abgegrenzt
- die Interoperabilität mit nicht .NET - Anwendungen wird gewährleistet
- der Type Checker prüft Datentypen und Typkonvertierungen
- der Class Loader lädt Klassen in die Laufzeitumgebung

3.2 Framework Class Library

Die .NET Framework Class Library(FCL)[8] stellt die Kernfunktionalitäten des .NET Framework dar. Diese sehr umfangreiche Klassenbibliothek stellt allen .NET - Sprachen eine Vielzahl Klassen, Schnittstellen und Strukturen zu Verfügung. In weiten Teilen stellt die FCL aber nur einen Wrapper für Funktionen aus der Win32 - Api oder bestehenden COM - Komponenten dar. Die FCL ist streng hierarchisch aufgebaut, die Oberklasse der Hierarchie bildet die Klasse System. Von dieser Klasse leiten sich alle weiteren Unterklassen ab. Die Unterklassen können wiederum über eigene Unterklassen verfügen.

Um die Übersicht zu gewährleisten sind die FCL - Klassen in 312 Namensräume gegliedert, diese gruppieren die Klassen logisch mit einander. Durch das Einbinden eines Namensraum zu Anfang einer Datei kann darauf verzichtet werden, den voll qualifizierenden Klassenname des verwendeten Objektes im Verlauf der Datei erneut anzugeben, denn der Klassenpfad kann so entfallen. Die Technik der Namensräume kann man sich auch in der Entwicklung zunutze machen, da Namensräume auch im Quelltext definiert werden können.

Innerhalb der FCL werden die fundamentalen Namensräume auch als .NET Base Library(BCL) bezeichnet. Dazu gehören folgende Teile des System Namensraum: *CodeDom, Collections, Diagnostics, Globalization, IO, Rsesources, Text, RegularExpressions.*

Im Folgenden werden einige wichtige Teile der Namensräume vorgestellt, die eigentlich eigenständige Technologien im .NET Framework darstellen und auch nicht in dem ECMA Standard zertifiziert sind. Aufgrund des Umfangs dieser Arbeit können nur exemplarisch einige Teilaspekte Erwähnung finden.[9]

---

[6] (Microsoft: Common Language Runtime)

[7] (Microsoft: Common Language Runtime Overview)

[8] (Schwichtenberg, Microsoft - .NET - 4.0 - Crashkurs, 2011) S. 127ff

[9] (Monadjemi, 2009)

3.2.1 User Interface

Um grafische Benutzeroberflächen zu implementieren sind die Namensräume zu drei großen Technologien zusammengefasst.[10]

Windows Forms

Windows Forms ist eine der beiden Bibliotheken für Desktop - Oberflächen im .NET Framework. Windows Forms wurde in .NET 1.0 eingeführt und wird seit .NET 3.0 nicht mehr weiterentwickelt. Die Windows Forms werden im Namensraum *System.Windows.Forms* bereitgestellt.

Windows Presentation Foundation

Die Windows Presentation Foundation(WPF) kann man als Nachfolger von Windows Forms verstehen, wobei es sich um eine vollkommene Neuentwicklung handelt, die mit einer Vielzahl von neuen Funktionen ausgestattet ist. Mit der WPF können verschiedene Arten von Oberflächen erstellt werden, die über klassische Desktop Anwendungen hinausgehen. Alle *System.Windows* Namensräume, die nicht Forms heißen, gehören zu WFP.

Im Gegensatz zu Windows Forms kann in der WPF die Oberfläche sowohl im Code als auch in der Application Markup Language(XAML), einem XML - Dialekt, definiert werden, was eine optionale Trennung von Layout und Inhalt ermöglicht.

Weitere Neuerungen sind:[11]

• Die Oberflächen können als Fenster, im Webbrowser und in einem eigenen Viewer dargestellt werden

• Die Anzeige ist vektorbasiert und lässt sich so gut skalieren

• 2D und 3D Unterstützung

• Abspielen von Ton und Videos

Asp.NET

Im Namensraum *System.Web* werden alle Klassen für Asp.NET bereitgestellt. Asp.NET kann als Weiterentwicklung von Active Server Pages(Asp) betrachtet werden und ermöglicht die Erstellung von Dynamischen Webseiten. Hierfür bietet Asp.NET eine ganze Reihe von Webcontrols an, deren Funktionalität die der HTML - Controls übersteigt. Zudem können eigene Controls, die in einer .NET - Sprache geschrieben wurden, auf die gleiche Art wie die Standard Controls eingebunden werden. Auch hier kann optional eine Trennung von Layout und Inhalt erreicht werden. Eine Asp.NET - Seite besteht aus einer .aspx Datei welche die Webcontrols enthält, und die einen Script - Block enthalten kann, in dem alle .NET - Sprachen verwendet werden können. Optional kann der Code in einer Code - Behind - Datei ausgelagert werden. Im Gegensatz zu dem alten Asp werden Asp.NET - Seiten vor der Veröffentlichung auf dem Webserver kompiliert und so dort gespeichert. Bei einer Anfrage der Asp.NET - Seite wird die Seite in den Arbeitsspeicher des Webserver geladen und generiert dann den

[10] (Chappell, 2002)

[11] (Wenger, 2010)

HTML - Code, der an den Browser ausgeliefert wird. So konnte die Ausführungsgeschwindigkeit im Vergleich zu den alten Asp - Webseiten deutlich erhöht werden.[12]

3.2.2 Services

Windows Communication Foundation

Die Windows Communication Foundation (WCF) stellt innerhalb des .NET Framework eine einheitliche API zur verteilten Kommunikation von Anwendungen bereit und kapselt so eine ganze Reihe von früheren Technologien. Das Hauptaugenmerk liegt hier auf der Entwicklung von Service - orientierten Architekturen(SOA).[13]

Windows Workflow Foundation

Die Windows Workflow Foundation ist eine API, die es ermöglicht komplexe Workflow in Software abzubilden, ohne diese direkt im Quellcode zu implementieren. Hiermit wird eine Trennung des Workflow von dem rechtlichen Programm erreicht, was spätere Anpassungen und die Wartung erleichtert.

3.2.3 Data Access

Im Namensraum *System.Data* befinden sich alle Klassen zu ADO.NET, die den Umgang mit Datenbanken erleichtern sollen. Der Name ADO.NET leitet sich von ActiveX Data Objects (ADO) ab, ist aber nicht mehr in ActiveX implementiert. Es bietet Klassen zur Anbindung an eine Vielzahl von relationalen Datenbanksystemen und Klassen um auf den relationalen Daten zu arbeiten. Mit den Entity Framework gibt es innerhalb von ADO.NET auch die Möglichkeit die relationalen Daten auf Objekte zu mappen.

3.3 Programmausführung

Der in einer .NET - Sprache geschriebenen Quellcode wird von einem sprachspezifischen Compiler in die Microsoft Intermediate Language(MSIL) überführt. MDIL ist Microsofts Bezeichnung für CIL. Das Ergebnis des kompilieren ist eine Assembly, also eine ausführbare Datei(.exe) oder eine Dynamic Link Library(DLL). Diese besteht aus dem Code, einer Manifest - Datei die alle Komponenten und Klassen in XML spezifiziert und einigen Metadaten. Microsoft unterscheidet zwischen Managed Code, der unter der Kontrolle der Laufzeitumgebung abläuft und Unmanaged Code oder Native Code. Jede Assembly besteht aus Managed Code, es kann aber auch teilweise Native Code enthalten.

[12] (Schwichtenberg & Fuchs, Webanwendungen mit ASP.NET 3.5 und Ajax, 2008)

[13] (Wenz, Heuser, Kotz, & Samaschke, 2011)

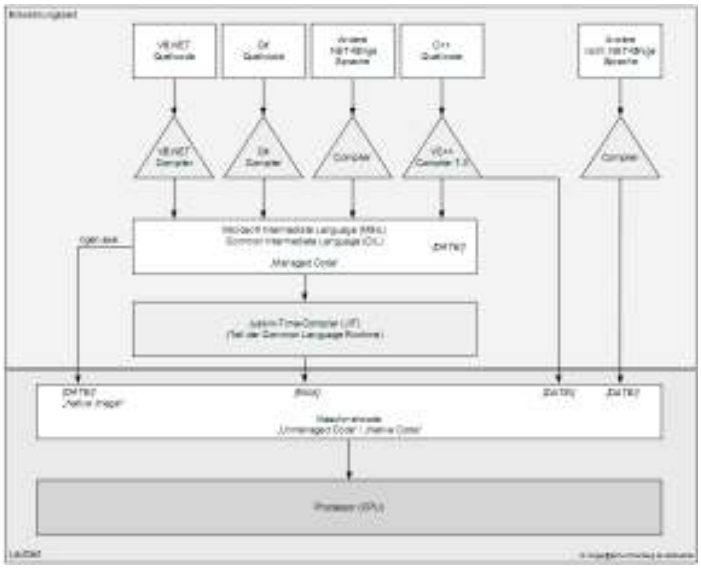

Abbildung 2: Zwischensprachkonzept[14]

Zur Laufzeit wird der MSIL - Code von einem Just - in - Time - Compiler optimiert und dann in prozessorspezifischen Native - Code umgewandelt und ausgeführt. Hierzu lädt der Class Loader, anhand des Assembly - Manifest, die zur Anwendung gehörenden Klassen. Da diese noch im MSIL - Code vorliegen, werden die Klassen auf Korrektheit, Typsicherheit und auf die richtigen Signaturen geprüft. Der Just - in - Time Compiler wandelt nun den gesamten Code in plattformspezifischen Nativ Code um. Dieser wird nun durch den Code Manager ausgeführt. Der Code Manager regelt dabei die Speicherbereinigung und die Ausnahmebehandlung. Alternativ besteht die Möglichkeit, den erzeugten Native - Code in einem sogenannten Native Image zu speichern und später direkt auszuführen, was die Ausführungsgeschwindigkeit deutlich erhöht.

Um eine .NET - Anwendung ausführen zu können, muss erst die CLR in den Speicher geladen werden und die Anwendung an die CLR übergeben werden. Dafür wird für jeden Typ von .NET - Anwendung ein eigener .NET - Runtime Host benötigt. Jeder Runtime Host besitzt am Anfang einen sogenannten Stub, der aus Unmanaged Code bestehen muss, da noch keine CLR geladen ist. Der Stub hat die Aufgabe, mit Hilfe der .NET Hosting API, die CLR in den Speicher zu laden und die Anwendung an die CLR zu übergeben.

3.4 Programmiersprachen

Microsoft .NET unterstützt eine Vielzahl von Programmiersprachen, die sowohl kommerziell als auch kostenlos angeboten werden. Darunter sind sowohl objektorientierte als auch funktionale Programmiersprachen. Microsoft liefert aktuell die Sprachen Visual Basic.NET, C#, F#, JScript .NET, Visual C++und IronPyton. Im Folgenden werden nur die von Microsoft angeboten Sprachen kurz vorgestellt, da diese am stärksten verbreitet sind.

[14] (Schwichtenberg, Microsoft - .NET 3.5 Crashkurs, 2008)S. 58

3.4.1 Visual Basic.NET

Visual Basic.NET wurde aus Visual Basic entwickelt, ist aber seit der Version 7.0 nicht mehr mit dieser kompatibel. Visual Basic galt lange als einfach und produktiv, aber auch als unsauber und unstrukturiert. Das hat sich mit der Version 7.0 geändert, denn Microsoft hat alle Sprachkomponenten, die nicht mit der .NET Framework zu vereinbaren waren, entfernt. Außerdem wurden vor allem Funktionen zum wirklichen Objektorientierten Programmieren hinzugefügt, die folgende Möglichkeiten bieten.

• Einfachvererbung, abstrakte Klassen, Interfaces, Operatorüberladung und Überschreiben von Methoden

• Parametrisierte Konstruktoren

• Strukturiertes Exception Handling

• Shared Members: Klassenmethoden und Klassenvariablen

3.4.2 C# (C Sharp)

C# ist eine speziell für .NET neuentwickelte Programmiersprache, die vor allem an C++ und Java angelehnt und unter dem ECMA - 334[15] zertifiziert ist. Hierbei wurden Bereiche ausgespart, die erfahrungsgemäß Probleme bereiten. Zu diesen Bereichen zählt zum Beispiel die Mehrfachvererbung in C++. In C# können, genau wie in Java, Klassen nur von einer Basisklasse erben. Darüber hinaus verzichtet C# auf den Gebrauch von Headerdateien. Stattdessen stehen Code und Deklaration in einer Datei. Eine Verteilung einer Klasse über mehrere Dateien, wie in C++, ist daher nicht möglich. Dies führt zu leichter verständlichen und pflegbareren Programmen. Eine weitere Fehlerquelle wurde entschärft, indem die Verwendung von Zeigern nur möglich ist, wenn der betreffende Code als unsafe deklariert wird. Die Verwendung von unsicherem Code hat zur Folge, dass dieser nicht mehr unter der Verwaltung der CLR ausgeführt wird. Der Code wird auch nicht mehr auf die Sicherheit von Datentypen oder Speicherzugriffen getestet und benötigt daher volle Zugriffsrechte. Verzichtet man aber auf unsicheren Code, dann sorgt die automatische Speicherverwaltung dafür, dass Zugriffe auf ungeschützte oder ungültige Speicherbereiche ausgeschlossen sind.

3.4.3 F# (F Sharp)

F# ist die funktionale Programmiersprache von Microsoft. Sie bietet aber auch objektorientierte Sprachkonstrukte an. Der Syntax F# ist OCaml und Ml sehr ähnlich, so dass es möglich ist, OCaml Code, der nicht auf OCaml eigene Bibliotheken zurückgereift, direkt mit F# zu kompilieren. Seit 2010 steht F#, samt der Klassenbibliothek und des Compiler, unter der Apache - 2.0 Lizenz. [16]

3.4.4 JScript.NET

JScript.NET ist aus der Sprache JScript entstanden und ist zu älteren Versionen von dieser Sprache vollständig abwärts kompatibel. Beide Sprachen sind JavaScript Derivate und zum Großteil mit JavaScript kompatibel. Doch Microsoft hat eigene Sprachkonstrukte eingeführt und dadurch unterstützt der Microsoft Internet Explorer, JScript.NET als einziger Browser komplett, wobei immer mehr Browser die Sprache implementieren.

---

[15] (ECMA - 334: C# Language Specification, 2006)

[16] (Microsoft F# Developer Center)

3.4.5 Visual C++

Weiterhin bietet Microsoft einen C++ - Compiler, der im Gegensatz zu allen anderen Microsoft .NET - Sprachen, neben managed Code auch direkt native Code erzeugen kann. Visual C++ ist auch nicht primär für die Einwicklung von .NET konzipiert. Es handelt sich dabei um normales C++, das durch Managed Extenstion erweitert wurde. Die Managed Extensions ermöglichen, dass auch hier managed und unmanaged Code mit einander kompatibel ist.

3.4.6 IronPython

IronPython ist die vollkommen in C# geschriebene Python Implementierung von Microsoft. Sie enthält zwar keine Python Standard Bibliothek, kann diese aber laden solange diese keinen vorkompilierten Code enthält. Mit Ausnahme dieser Einschränkung ist IronPython sonst vollständig mit Python 2.7 kompatibel. Auf die .NET Klassen Bibliotheken kann IronPython uneingeschränkt zugreifen, sowie andere .NET Sprachen mit kleinen Einschränkungen auf IronPython Bibliotheken zugreifen können. Wie viele andere .NET Sprachen wird sie unter der Apache Lizenz 2.0 bereitgestellt.

3.5 Microsoft Software Komponenten

Microsoft bietet eine Reihe von Softwareprodukten an, die das Entwickeln im .NET Framework erleichtern sollen. Dazu gehört in erster Linie die Entwicklungsumgebung Visual Studio, die neben zahlreichen grafischen Designer zur GUI Erstellung und Datenbank Anbindung, über eine Autovervollständigung des Quellcode verfügt. Für die grafische Entwicklung steht das Expression Studio zu Verfügung. Expression Studio besteht aus mehreren Programmen, die für Webdesign oder auch zur Bearbeitung der WPF Oberflächen gedacht sind. Darüber hinaus gibt es noch eine große Anzahl weiterer Programme.

3.6 Microsoft Server Infrastruktur

Die Microsoft Server Betriebssysteme bilden mit ihren Funktionalitäten die Grundlage für viele .NET Programme. Der Internet Information Services(IIS) ist ein Web Server für Microsoft Server, der vor allem auf Asp.NET Applikationen zugeschnitten ist. Der Microsoft SQL Server ist ein relationales Datenbankmanagementsystem und leicht über verschiedene Techniken in .NET Programme zu integrieren. Es gibt noch eine ganze Reihe von Serverprodukten die Microsoft auf das .NET Framework ausgerichtet hat.

4. Mono

Mono ist eine Open - Source Implementierung des ECMA Standards von CLI und steht unter GNU Lesser General Public License (LGPL). Miguel de Icaza, eine Mitbegründer von GNOME, entwickelte mit seinem Unternehmen XIMIAN[17] die freie .NET Implementierung unter LGPL und gründete das

[17] (Golem: Miguel de Icaza gründet Mono - Unternehmen)

Mono Projekt, das die Weiterentwicklung von Mono betreibt. Novel übernahm XIMIAN und lies Mono weiter unter LGPL bestehen und unterstützt das Mono Projekt bei der Erweiterung.[18]

4.1 Lizenz

Insgesamt steht Mono unter einer Kombination von drei Lizenzen. Der C# - Compiler steht unter MIT/X11 und General Public License(GPL) und alle Tools nur unter GPL. Die Laufzeitumgebung steht unter LGPL und die Klassenbibliotheken nur unter MIT/X11. Der Großteil, der nicht in dem EMAC Standard enthaltenen Klassen, steht unter der Microsoft Permissive License und einige auch unter Apache 2 Lizenz. Die *Mono Tools for Visual Studio Ultimate Edition* enthalten auch eine Kommerzielle Lizenz. Diese ist für Projekte gedacht, die gegen die LGPL Lizenz verstoßen und deren Umsatz unter zwei Millionen Dollar pro Jahr liegen. [19]

4.2 Plattformunabhängigkeit

Mono ist vornehmlich entwickelt worden um das .NET Framework auf Unix basierten Betriebssysteme wie Linux einsetzen zu können. Heute läuft Mono auf vielen verschiedenen Systemen, wie Linux, Microsoft Windows, Mac OS X, BSD, Sun Solaris, Nintendo Wii, Sony PlayStation 3, Apple iPhone und Googel Android, und somit auch auf den unterschiedlichsten Rechnerarchitekturen.[20] Mono ist in die Suse und Debian Standardinstallation integriert. Fedora hingegen entfernte Mono schon nach einer Version wieder aus seiner Standardinstallation.

4.3 Kompatibilität

Mono implementiert einen Großteil der .NET Base Class Library und ist auch mit dieser kompatibel. Über den ECMA Standards hinaus unterstützt Mono auch Asp.NET, ADO.NET und Windows Forms. Andere Elemente wie die WPF und die WCF werden aber voraussichtlich nicht unterstützt oder, wie WF, erst in kommenden Versionen eingeführt.

Mono enthält darüber hinaus eine Vielzahl weiterer Klassen, vor allem um die Entwicklung für Linux zu erleichtern, wie beispielsweise OpenGL und POSIX.[21]

4.4 Sprachen

In Mono kann in jeder Programmiersprachen, die einen vollständig CIL kompatiblen Code erzeugt, entwickelt werden. Das Mono Projekt beinhaltet Compiler für Microsoft C#, Basic und Ilasm, wobei

[18] (Diedrich)

[19] (Mono: Licensing)

[20] (Mono: What is Mono)

[21] (Mono: Compatibility)

C# die primäre Programmiersprache in Mono ist. Eine weitere Besonderheit ist die Java Unterstützung, die Mithilfe von IKVM, einer in .NET geschrieben Java - virtual Maschine, erreicht wird. IKVM implementiert die Java Klassenbibliothek in .NET und stellt Operationen zur Interaktion zwischen Java und .NET Klassen bereit.[22]

4.5 Infrastruktur

Für die Entwicklung und das Hosten von Asp.NET Applikationen gibt es einige Server die Mono unterstützen. Zu erwähnen wären vor allem Mod_Mono als Appache Modul und der Webserver Nginx. Das Mono Projekt bietet MonoDevelop als integrierte Entwicklungsumgebung an, dies ist aber nicht die einzige freie Entwicklungsumgebung die mit Mono kompatibel ist. Mit Mono Migration Analyzer wir auch ein Tool zur Überführung von .NET Applikationen in Mono angeboten.[23]

4.6 Exkurs DotGNU

DotGNU ist die zweite freie .NET Implementierung, wobei anfänglich der Schwerpunkt auf die Entwicklung von Webservices und Applikationen in C# gelegt wurde. Der Funktionsumfang ist mit der Zeit deutlich gewachsen, so dass auch VB.NET und einige nicht standardisierte Teile von .NET unterstützt werden. Mittlerweile werden neben Linux, auch MS Windows und Apple OS unterstützt.[24]

Die drei Hauptkomponenten von DotGNU bestehen aus:

Portable.NET

Hier werden die CLI implementiert und es werden Programme zum kompilieren von VB.NET, C# und C bereitgestellt.

phpGroupWare

Dieser Teil implementiert eine webbasierte GroupWare Suite, die eine Sammlung von Webservice – Komponenten bereitstellt.

DotGNU Execution Environment

Ein speziell auf DotGNU zugeschnittener Webserver.

Alle Programme und der Beispielquelltext stehen unter GNU General Public License(GPL) und alle Bibliotheken unter LGPL. 2006 wurden Stimmen laut, dass die Weiterentwicklung des Projekts ruht.

[22] (Mono: What is Mono)

[23] (Mono: Mod mono), (Mono: FastCGI Nginx), (Mono: MonoDevelop), (SharpDevelop)

[24] (DotGNU Project)

Mono scheint DotGNU den Rang abgelaufen zu haben, denn auch die letzten offiziellen Bekanntgaben enden 2009. [25]

4.7 Kritik an Mono

Gerade in der Open Source Welt wurde immer wieder davor gewarnt freie .Net Implementierungen in der Softwareentwicklung einzusetzen. Es wurde die Befürchtung geäußert, dass Microsoft wegen Patentverletzung gegen Mono oder dotGNU vorgehen könnte. Aus dem gleichen Grund wurde auch davor gewarnt Programme, die in Mono oder dotGNU geschrieben wurden, in die Standardinstallation diverser Linux Distributionen aufzunehmen.[26]

Um diesen Befürchtungen entgegen zu treten, stellte Microsoft die beiden Teile des .NET - Framework, die dem ECMA Standard 334 und 335 zuzuordnen sind, unter die Microsoft Community Promise. Spezifikationen die unter der Community Promise stehen, könne von jedem implementiert und verwendet werden und die daraus entstanden Technologien können, ohne jede Lizenzbeschränkung, verkauft werden. Weiter verspricht Microsoft nicht gegen die freien .NET Spezifikationen vorzugehen.

Dies geht einigen Kritikern nicht weit genug, da die Lage um die Komponenten von .NET, die nicht standardisiert sind, weiter unklar bleibt.[27]

4.8 Die Zukunft von Mono

Am 20.04.2011 hat Attachmate Novel übernommen, dies geschah mit Hilfe der neu gegründeten CPTN Holdings, die wie sich später herausstellte ein Konsortium von Microsoft, Apple, Oracle und EMC ist. Die CPTN Holdings sicherte sich dabei 882 Patente von Novel, und Microsoft war angeblich auch an der Finanzierung der restlichen Novel Übernahme beteiligt. Das US Justizministerium hat den Patentverkauf an die CPTN Holdings unter seht strengen Auflagen mittlerweile zugestimmt. Die Auflagen sehen vor, dass alle Patente unter GPL und unter der Lizenzvereinbarung des Open Invention Network (OIN) gestellt werden müssen und sie regeln auch die Aufteilung der Patente unter den beteiligten Unternehmen.[28]

Direkt nach der Übernahme entließ Attachmate einen Großteil seiner Mono Entwickler und Mono wurde dem Geschäftsbereich von SUSE Linux in Nürnberg zugeordnet. Seitdem überschlagen sich die Ereignisse um Mono.[29]

---

[25] (Linux Magazin)

[26] (Mono: What is Mono), (Free Software Foundation)

[27] (Open Souce Community at Microsoft), (Peter Galli), (LWN.net)

[28] (Golem: Novell-Patente), (Linux Verband)

[29] (ZDNet: Attachmate feuert Novells Mono-Entwickler)

Der Datenbankhersteller Sones hat beispielsweise angekündigt, die Mono - Foundation als Stiftung für die Entwickler des Mono - Projekt ins Leben zu rufen und die Weiterentwicklung vorantreiben zu wollen. Doch die Zukunft von Mono ist momentan ungewiss. [30]

5. Fazit

Zusammenfassend lässt sich sagen, dass Microsoft mit den .NET Framework, Softwareentwicklern eine sehr mächtige Infrastruktur bietet. Vor allem weil das Framework seit Windows Vista direkt mit dem Betriebssystem ausgeliefert wird. Damit kann jede .NET Anwendung sofort auf einem neueren Windowssystem ausgeführt werden, ohne dass irgendeine Bibliothek installiert werden muss.

Microsoft hat sich im .NET Framework stark von Java inspirieren lassen, aber auch eigene interessante Konzepte beigesteuert. Microsoft .NET bietet für eine Vielzahl von Szenarien die Möglichkeit, auf ein und dasselbe Framework zurück zu greifen, indem immer stärkere Programmierprinzipien wie die Trennung von Layout und Funktion Anwendung finden. Auch geht in .NET der Trend immer stärker zur komponentenbasierten Entwicklung, die eine noch stärkere Abstraktion von der eigentlichen Programmlogik propagiert.

Mono macht die Entwicklung von Anwendungen in diesem Bereich noch interessanter. Vor allem weil mittlerweile große Teile des .NET Framework in Mono integriert sind. Was sich am stärksten in dem nicht standardisierten Teil von .NET äußert. Der größte Vorteil von Mono ist jedoch die wirkliche Plattformunabhängigkeit. Diese ermöglicht es auch für Plattformen, Software zu entwickeln die von Microsoft nicht unterstütz werden, was sich zur Zeit vor allem in der App - Entwicklung äußert. Microsoft .NET steht nur für Windows Phone 7 zu Verfügung. Durch Mono ist es auch möglich für iOS und Android zu entwickeln. Dies bietet eine gute Möglichkeit den Quellcode zwischen den beiden Plattformen zu portieren.

Abzuwarten bleibt, wie sich die Situation um Mono weiter entwickelt und wie sich die Aufteilung der Novel Patente auswirkt. Fragwürdig ist die Rolle die Microsoft bei der Novel Übernahme gespielt hat. Die Befürchtungen der Mono Kritiker haben sich so bewahrheitet. Welche Folgen dieses Vorgehen für die freie Softwareentwicklung hat, ist heute noch nicht abzusehen.

[30] (Golem: Stiftung soll Entwickler betreuen)

Literaturverzeichnis

.NET Framework 4.0. Abgerufen am 30. 04 2011 von http://blogs.microsoft.co.il/blogs/ohad/

Chappell, D. (2002). .Net verstehen. München: Addison-Wesley.

Diedrich, O. Heise: Was die Kunden wollen Microsoft und Novell kooperieren. Abgerufen am 13. 06 2011 von http://www.heise.de/open/artikel/Microsoft-und-Novell-kooperieren-222001.html?view=print

DotGNU Project. Abgerufen am 02. 07 2011 von www.gnu.org/software/dotgnu/

ECMA-334: C# Language Specification. (2006). Abgerufen am 17. 05 2011 von http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf

ECMA-335: Common Language Infrastructure . (2010). Abgerufen am 17. 05 2011 von http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-335.pdf

Free Software Foundation. Abgerufen am 02. 07 2011 von http://www.fsf.org/news/dont-depend-on-mono

Golem: Miguel de Icaza gründet Mono-Unternehmen. Abgerufen am 27. 05 2011 von http://www.golem.de/1105/83536.html

Golem: Novell-Patente. Abgerufen am 06. 07 2011 von http://www.golem.de/1104/82969.html

Golem: Stiftung soll Entwickler betreuen. Abgerufen am 06. 07 2011 von http://www.golem.de/1106/84027.html

Linux Magazin. Abgerufen am 05. 07 2011 von Die freien Dotnet-Implementierungen Mono und DotGNU im Vergleich: http://www.linux-magazin.de/Heft-Abo/Ausgaben/2005/10/Ueberall-Punkte

Linux Verband. Von http://www.linux-verband.de/news/detail/opensource/novell-geht-an-attachmate/ abgerufen

LWN.net. Von How Mono got into Fedora: http://lwn.net/Articles/179597/ abgerufen

Microsoft Community Promise. Abgerufen am 13. 06 2011 von http://www.microsoft.com/openspecifications/en/us/programs/community-promise/default.aspx

Microsoft F# Developer Center. Abgerufen am 03. 05 2011 von http://msdn.microsoft.com/en-us/fsharp/default

Microsoft Reference Source License. Abgerufen am 20. 06 2011 von http://referencesource.microsoft.com/referencesourcelicensing.aspx

Microsoft: .NET Framework Class Library Overview. Abgerufen am 17. 05 2011 von http://msdn.microsoft.com/en-us/library/hfa3fa08.aspx

Microsoft: Common Language Runtime. Abgerufen am 15. 05 2011 von http://msdn.microsoft.com/en-us/library/8bs2ecf4%28v=VS.71%29.aspx

Microsoft: Common Language Runtime Overview. Abgerufen am 17. 05 2011 von http://msdn.microsoft.com/en-us/library/ddk909ch%28v=VS.71%29.aspx

Microsoft: Konzeptionelle Übersicht über das .NET Framework. Abgerufen am 01. 05 2011 von http://msdn.microsoft.com/de-de/library/zw4w595w.aspx

Microsoft: Managed Execution Process. Abgerufen am 31. 05 2011 von http://msdn.microsoft.com/en-us/library/k5532s8a%28v=VS.71%29.aspx

Monadjemi, P. (2009). Visual Basic 2008: Windows-Programmierung mit Visual Basic 9.0, Visual Studio 2008 und .NET Framework 3.5 . München: Markt-&-Technik-Verl. .

Mono: Compatibility. Abgerufen am 01. 05 2011 von http://mono-project.com/Compatibility

Mono: FastCGI Nginx. Abgerufen am 25. 05 2011 von http://mono-project.com/FastCGI_Nginx

Mono: Licensing. Abgerufen am 31. 06 2011 von http://www.mono-project.com/Licensing

Mono: Mod mono. Abgerufen am 25. 05 2011 von http://mono-project.com/Mod_mono

Mono: MonoDevelop. Abgerufen am 25. 05 2011 von http://monodevelop.com/

Mono: What is Mono. Abgerufen am 01. 05 2011 von http://mono-project.com/What_is_Mono

Open Souce Community at Microsoft. Abgerufen am 01. 07 2011 von http://blogs.technet.com/b/port25/archive/2009/07/06/the-ecma-c-and-cli-standards.aspx

Peter Galli. Abgerufen am 02. 07 2011 von Microsofts Open-Source-Blog: http://blogs.technet.com/b/port25/archive/2009/07/06/the-ecma-c-and-cli-standards.aspx

Schwichtenberg, H. (2008). Microsoft-.NET 3.5 Crashkurs. Unterschleißheim: Microsoft Press.

Schwichtenberg, H. (2011). Microsoft-.NET-4.0-Crashkurs. Unterschleißheim: Microsoft Press.

Schwichtenberg, H., & Fuchs, J. (2008). Webanwendungen mit ASP.Net 3.5 und Ajax. Unterschleißheim: Microsoft Press.

SharpDevelop. Abgerufen am 28. 05 2011 von http://sharpdevelop.net/opensource/sd/

Wenger, R. (2010). Handbuch der .Net 4.0 Programmierung. Köln: O`Reilly.

Wenz, C., Heuser, T., Kotz, J., & Samaschke, K. (2011). ASP.Net 4.0 mit Visual Basic 2010. München: Addison-Wesley.

ZDNet: Attachmate feuert Novells Mono-Entwickler. Abgerufen am 06. 07 2011 von http://www.zdnet.de/news/41552599/attachmate-feuert-novells-mono-entwickler.htm