Entwicklungstechnologien - .NET Compact Framework

Hausarbeit zum Forschungsseminar
„Mensch-Maschine-Interaktion:
Interaktion mit Location Based Services“

Walter Kern
25.08.2008

Gliederung

1 Einleitung

Die Zielsetzung dieser Arbeit, die im Rahmen des Forschungsseminars Mensch-Maschine-Interaktion mit dem Schwerpunkt „Location Based Services“ erstellt wurde, ist die Untersuchung der Entwicklungstechnologie Microsoft .NET Compact Framework mit dem Einsatzbereich mobiler Geräte.

Wenngleich die Primärintention dieser Arbeit darin besteht, einen Überblick über das .NET Compact Framework in seiner Gesamtheit zu geben, soll an geeigneter Stelle auf die Spezifika dieses Frameworks im Hinblick auf ortsabhängige Dienste, sogenannte Location Based Services, eingegangen werden.

Um den Hintergrund dieser Arbeit zu verdeutlichen und eine Basis für das Verständnis der in dieser Arbeit erörterten Punkte zu schaffen, soll zunächst der zentrale Begriff der „Location Based Services“ definiert und im Anschluss daran die zunehmende Bedeutung in diesem Kontext relevanter Technologien aufgezeigt werden.

1.1 Der Begriff „Location Based Services”

Für den Begriff Location Based Services (LBS), der auch als ortsabhängige Dienste übersetzt werden kann, gibt es nach Küpper (2005: 1ff.) trotz der Existenz des Begriffs seit etlichen Jahren keine allgemeine Definition oder Terminologie, was zum Teil an den unterschiedlichen Sichtweisen verschiedener Akteure wie dem Telekommunikationssektor oder dem Ubiquitous Computing-Bereich liegt.

So definiert die GSM Association[1] nach ebd. LBS abstrakt als Dienste, die die Position einer zu lokalisierenden Entität dazu benutzen, um einen Dienst mit zusätzlichem Nutzwert anzureichern. Als Beispiele für entsprechende Nutzwerte werden ortsabhängige Informationsfilterung im Kontext von nahegelegenen Points of Interest, Visualisierung eines Zielorts oder automatische Dienstaktivierung bei Passieren definierter Orte angeführt.

Daneben besagt nach ebd. eine Definition des 3rd Generation Partnership Project[2] (3GPP), wobei es sich um eine internationale Vereinigung vieler nationaler Standardisierungsbehörden handelt, welche die Bereitstellung von GSM[3]- und UMTS[4]-Spezifikationen als Zielsetzung besitzen, dass LBS ein Dienst eines Service-Providers ist, der die Lokationsinformation eines Terminals nutzt.

Nach Küpper (2005: 2f.) und Schmidt und Laerhoven (2001) werden Location Based Services im wissenschaftlichen Umfeld als Teilmenge der Context Aware Services gesehen. Dabei handelt es sich um Dienste, die sich in ihrem Verhalten auf Basis von Parametern, die den jeweiligen Kontext ausdrücken und als Kontextinformation bezeichnet werden, automatisch anpassen. Die Kontextinformation ist dabei im Fall der Location Based Services die jeweilige Ortsinformation.

[1] Konsortium, welches nach GSM Association (2008) aus über 750 GSM-Netzbetreibern besteht; unter Anderem Orange, China Mobile und T-Mobile; im Internet: http://www.gsmworld.com/index.shtml.
[2] Im Internet: http://www.3gpp.org/.
[3] GSM = Global System for Mobile Communications; Mobilfunkstandard der 2. Generation.
[4] UMTS = Universal Mobile Telecommunications System; Mobilfunkstandard der 3. Generation.

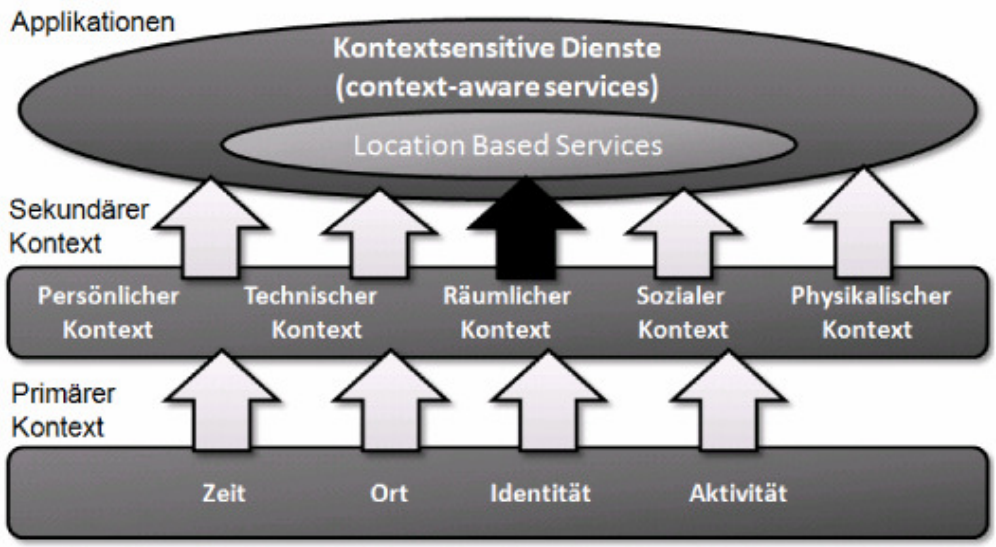

Abbildung 1: Kontextsensitivität und Location Based Services nach Küpper (2005: 2f.)

Wie in Abbildung 1 dargestellt, lässt sich eine weitere Unterteilung des Gesamtkontexts vornehmen, wobei der dargestellte primäre Kontext nach Küpper (2005: 2f.) eine Entsprechung des von Schmidt und Laerhoven (2001) benannten Cue Tuple Space samt zugehörigem Kontext darstellt und die Aufnahme der Rohdaten, z.B. über Sensoren, beschreibt. Der sekundäre Kontext ist hingegen dem in ebd. beschriebenen Context Tuple Space samt zugehörigem Kontext gleichzusetzen und stellt eine höhere Abstraktionsschicht zur Betrachtung der Daten, z.B. durch Kombination der Rohdaten des Primärkontexts, dar.

Als Beispiel für eine Abbildung eines ortsabhängigen Dienstes auf das von Küpper skizzierte Modell lässt sich ein Dienst für mobile Geräte ersinnen, welcher es den Benutzern ermöglicht, Freunde in der näheren Umgebung zu erkennen und gegebenenfalls automatisierte Aktionen, wie zum Beispiel Hinweissignale, bei Annäherung der Freunde zu emittieren. Primärkontexte wären in diesem Szenario der Ort, aber auch die Identität des Dienstanwenders. Als Sekundärkontext wäre vor allem der soziale Kontext hervorzuheben, der durch diese Visualisierung gesellschaftlicher Beziehungen im lokalen Umfeld ausgedrückt wird.

Ferner können nach Küpper (2005: 3) Location Based Services in reaktive und proaktive Dienste unterteilt werden. Ein reaktiver Dienst wird dabei stets vom Benutzer aktiviert (Pull-Prinzip), während ein proaktiver bei bestimmten Ereignissen, ohne Auslösung durch den Anwender, aktiv wird, z.B. bei Passieren eines bestimmten Ortes.

1.2 Die Bedeutung standortbezogener Dienste

Nach IDG Business Media GmbH (2005) sagt die Gartner-Prognose zur Entwicklung des Marktes für mobile Dienstleistungen eine Verdopplung des Marktes bis 2010 voraus. Neben dem Hardware- und Softwaresektor soll sich dies auch auf den Dienstleistungssektor auswirken.

Nach Hubschneider und Kölmel (2002: 5) handelt es sich bei Location Based Services um eine Killerapplikation des Mobile Business. Dies begründen Hubschneider und Kölmel mit den Ergebnissen verschiedener Studien zur erwarteten Entwicklung ortsabhängiger Dienste.

So kommt eine europaweite Umfrage der MediaTransfer AG Netresearch & Consulting zu dem Schluss, dass auf den Standort eines mobilen Nutzers zugeschnittene Dienste dem mobilen elektronischen Handel zum Durchbruch verhelfen werden (vgl. ebd.: 6).

Auch das Marktforschungsinstitut Mori sieht nach ebd. ein großes Marktpotential für LBS, was in Zahlen geschätzt bis zu 2,5 Milliarden Euro jährlich in Deutschland bedeutet. Aus diesem Grund soll im Folgenden die Entwicklungstechnologie .NET Compact Framework, mit deren Hilfe LBS realisiert werden können, untersucht werden.

2 Überblick über das .NET Compact Framework

In diesem Kapitel sollen die Grundlagen der Entwicklungstechnologie .NET Compact Framework behandelt werden. Dies umfasst sowohl eine Betrachtung des Mutter-Frameworks .NET-Framework als auch eine Untersuchung der Architektur des .NET Compact Framework selbst, sowie die Reflexion der Versionsgeschichte.

2.1 Das .NET Framework

Beim .NET Framework handelt es sich nach Troelsen (2007: 6) sowohl um eine Laufzeitumgebung als auch um eine äußerst umfangreiche Klassenbibliothek.

Als Zielplattformen kommen dabei für .NET-basierte Anwendungen neben Windows-Betriebssystemen auch Nicht-Microsoft-Betriebssysteme wie Mac OS X, Unix und Linux in Betracht (vgl. ebd.). Dies wird dadurch erreicht, dass die Kernbestandteile des .NET-Frameworks auf rechtlicher Ebene in Form von offenen ECMA[5]-Spezifikationen veröffentlich worden sind, was die Erstellung von weiteren Implementierungen wie Mono[6] neben der Microsoft-Referenzimplementierung zur Folge hatte (vgl. ebd.: 32f.).

Im Gegensatz zu Java (vgl. Louis und Müller, 2008), welches eine ähnliche Architektur wie das .NET-Framework aufweist (vgl. Manzoor, 2002), bestand bei .NET jedoch von Anfang an die Intention, mehrere Sprachen gleichberechtigt zu unterstützen (vgl. Troelsen, 2007: 9), weshalb neben den standardmäßig ausgelieferten Sprachen C#, Visual Basic .NET, J# und C++/CLI eine Vielzahl weiterer Sprachen .NET-tauglich gemacht worden ist[7]. Dies wird, wie in Abbildung 2 dargestellt, durch die Einführung einer Zwischensprache[8] erreicht. Sämtliche Anwendungen in einer beliebigen .NET-Sprache werden in diese Zwischensprache übersetzt, welche wiederum von der Common Language Runtime unter Zuhilfenahme des JIT[9]-Compilers bei Bedarf in nativen, prozessorspezifischen Code übersetzt wird.

Einsatzbereiche für das .NET-Framework sind neben der Desktopapplikations-, Dienste-, Konsolenanwendungs- und Webentwicklung[10] auch Scripting[11] und mobile Plattformen,

[5] ECMA = European Computer Manufacturers Association; internationale Normierungsorganisation für Informations- und Kommunikationstechnologien und die Unterhaltungselektronik.
[6] Im Internet: www.mono-project.com.
[7] Übersichtsliste über derzeit für die .NET-Plattform verfügbare Sprachen: http://www.dotnetlanguages.net.
[8] Die Common Intermediate Language, kurz CIL oder IL.
[9] JIT-Compiler = Just in Time Compiler; Maschinenunabhängiger IL-Code bzw. Bytecode wird erst bei Bedarf in prozessorspezifischen, nativen Code durch Kompilierung umgesetzt.
[10] Bei der Webentwicklung wird das ASP.NET-Framework (vgl. Esposito, 2005) eingesetzt, welches fester Bestandteil des regulären .NET-Frameworks ist.

wenngleich hierfür eine spezielle Variante, das .NET Compact Framework, bereitgestellt wird.

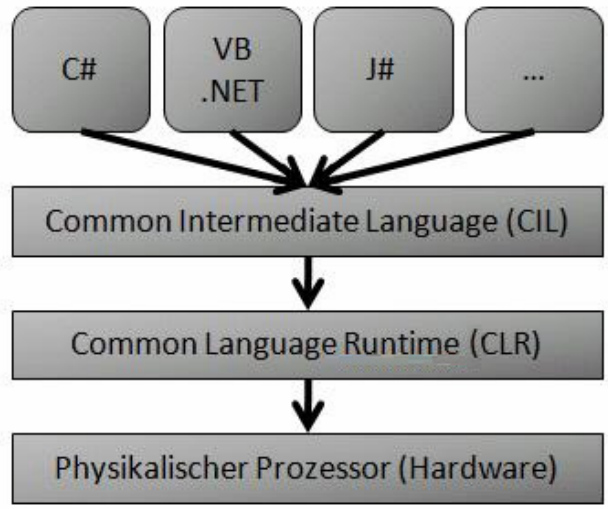

Abbildung 2: Vereinfachte Architekturdarstellung der .NET-Plattform

2.2 Das .NET Compact Framework

Das .NET Compact Framework, kurz .NET CF, ist eine spezielle Version des unter Punkt 2.1 erläuterten .NET Framework und zielt auf die Anwendungsentwicklung für mobile Endgeräte wie Pocket PCs, Smartphones, PDAs und eingebettete Systeme ab (vgl. Wigley et al., 2003: 17f. / Celarier und Creek, 2005). Dies schließt nach Microsoft Corporation (o.J.a), selbst Spielekonsolen wie die Xbox 360 durch eine spezielle Version des .NET Compact Framework mit ein.

2.2.1 Geschichte

Die erste Version des .NET Compact Framework mit der Versionsnummer 1.0 wurde nach Kuhbach (2003) im Jahr 2003 veröffentlicht. Zu dieser Version folgten noch mehrere Service Packs bis schließlich im Jahr 2005 das letzte Service Pack SP3 (vgl. Wells, 2005) erschien. Version 1.0 gibt dabei die grundsätzliche Architektur vor, die auch für die folgenden Versionen nahezu unverändert weiterverwendet worden ist. Zu den wesentlichen Bestandteilen des .NET Compact Framework zählen dabei nach Rubin und Yates (2003: 29f.) die Laufzeitumgebung Common Language Runtime und die .NET Compact Framework-Bibliotheken, welche es gemeinsam ermöglichen, Anwendungen in verwaltetem .NET-Code, auch Managed Code genannt, zu entwickeln und auf mobilen Plattformen durch eine Transformation des Managed Codes in Bytecode und schließlich nativen Code auszuführen (vgl. Kapitel 2.2.2).

Die nach WindowsForDevices.com (2005) im Jahr 2005 veröffentlichte Version 2.0 des Frameworks bringt nach Wilson (2005a) eine Vielzahl von Verbesserungen mit sich. Im Hinblick auf die Benutzeroberfläche werden einige neue reichhaltige Komponenten wie

[11] Automatisieren von Prozessen durch interpretierbare Sammlungen von Codeanweisungen. Beispielsweise basiert die Windows PowerShell auf dem .NET-Framework und unterstützt .NET-basiertes Scripting (vgl. Watt, 2007).

DateTimePicker und MonthCalendar, WebBrowser und benutzerdefinierte Steuerelemente unterstützt. Im Bereich Layout Management wurden nach ebd. die von Windows Forms für Desktopanwendungen bekannten Docking- und Anchor-Mechanismen auf den mobilen Anwendungsbereich abgebildet. Ferner wurde die Unterstützung im Bereich Daten hinsichtlich der Technologien XML und Data Binding[12] erheblich ausgebaut. Im Bereich Kommunikation wurde nach ebd. die Unterstützung von Webservices, MSMQ[13] und IPv6[14] verbessert bzw. implementiert. Neben der Erweiterung um Sicherheitstechnologien wurden auch die Performance und das Threadingverhalten optimiert. Schließlich wurde 2007 nach Dot-NET Compact Framework Team (2007) das bisher letzte Service Pack 2 veröffentlicht.

Ebenfalls im Jahr 2007 wurde mit Erscheinen des .NET Framework 3.5 und Visual Studio 2008 (vgl. dotnet.de, 2007) auch die neue und zum jetzigen Zeitpunkt aktuelle Version 3.5 des .NET Compact Framework veröffentlicht. Diese bietet nach Hart (2007) Erweiterungen im Bereich Multimedia, beispielsweise die einfache Möglichkeit zum Abspielen von wav-Dateien. Als eine der größten Neuerungen sind jedoch nach ebd. die Unterstützung von LINQ[15] und WCF[16] zu nennen. Damit wird zum Einen das typisierte und einfache Suchen und Filtern in komplexen Datenstrukturen und Datenquellen ermöglicht und zum Anderen die Anbindung von SOA[17]-Diensten auf eine deklarative und zugleich hochgenerische Art und Weise unterstützt.

2.2.2 Architektur

Das .NET Compact Framework besitzt grundsätzlich einen ähnlichen Aufbau wie das .NET Framework, welches unter Kapitel 2.1 näher beleuchtet wird. Dies schließt unter Anderem die theoretische Unterstützung beliebiger .NET-Sprachen und die Existenz einer Laufzeitumgebung, der Compact CLR, sowie die Bereitstellung eines Basisklassenframeworks mit ein. Es gibt nach Rubin und Yates (2003: 33f.) jedoch auch gravierende Unterschiede. So setzt das .NET CF beispielsweise auf einer Hardwareabstraktionsschicht (PAL) auf. Auch wird kompilierter Code beim .NET Compact Framework im Gegensatz zum .NET Framework nur für die Laufzeit einer Anwendung zwischengespeichert und nicht darüber hinaus.

2.2.2.1 Die Common Language Runtime

Wie in Abbildung 3, welche an Figure 2-1 der Microsoft .NET Compact Framework Core Reference (vgl. Wigley et al., 2003: 40f.) angelehnt ist, dargestellt, besteht der zentrale Teil des .NET CF, die Common Language Runtime (CLR), aus einem verwalteten und einem nativen Code-Teil.

Aufgabe der Common Language Runtime ist nach ebd.

- die Entgegennahme einer im MSIL[18]-Format vorliegenden Assembly[19],

[12] Data Binding = Automatische Synchronisation zwischen Benutzeroberflächenkomponenten und Datenquellen durch deklarative oder programmatische Zuordnungen.
[13] MSMQ = Microsoft Message Queuing; eine Nachrichtenwarteschlange, u.a. für netzübergreifende, asynchrone Kommunikation.
[14] IPv6 = Nachfolgeversion des heutzutage verbreiteten Internet Protocol der Version 4 zur Adressierung von Netzwerksystemen.
[15] LINQ = Language Integrated Query; vgl. Pialorsi und Russo (2007).
[16] WCF = Windows Communication Foundation; vgl. Kuhrmann und Beneken (2006).
[17] SOA = Serviceorientierte Architektur.
[18] MSIL = Microsoft Intermediate Language; wurde in Common Intermediate Language (CIL) umbenannt.

- die Bereitstellung einer Application Domain für die Ausführung (vgl. Kapitel 2.2.2.2),
- die Kompilierung des verwalteten Code in nativen Code um eine Ausführung auf dem jeweiligen Host-Prozessor zu ermöglichen,
- sowie die Verwaltung des Speichers, die Garbage Collection (vgl. Kapitel 2.2.2.4), der Klassenlademechanismus und die Implementierung von Sicherheit.

Realisiert werden die genannten Funktionalitäten durch das in Abbildung 3 dargestellte CLR-Schichtenmodell.

Anwendungen werden in einer beliebigen .NET-Programmiersprache verfasst und wie alle gerätespezifischen und benutzerdefinierten Bibliotheken sowie die Basisklassenbibliotheken selbst in MSIL-Code umgesetzt, welcher von der Compact CLR unter Zuhilfenahme des JIT-Compilers und der damit verbundenen Umwandlung in nativen Code ausgeführt wird. Dienste wie Speichermanagement und Garbage Collection werden durch die CLR bereitgestellt, weshalb auf der CLR ausgeführter Code auch als verwalteter Code bezeichnet wird (vgl. Wigley et al., 2003: 40). Im Gegensatz dazu wird Code, der außerhalb der Kontrolle der CLR ausgeführt wird, als unverwalteter oder nativer Code bezeichnet.

Nach ebd. bewirkt die Execution Engine (EE) die Ausführung von im MSIL-Code vorliegenden Anweisungen und ist darüberhinaus selbst eine native Anwendung. Der ebenfalls im nativen Code vorliegende Platform Adaption Layer (PAL) stellt eine Abstraktionsschicht zwischen der Execution Engine und der Betriebssystemschicht dar.

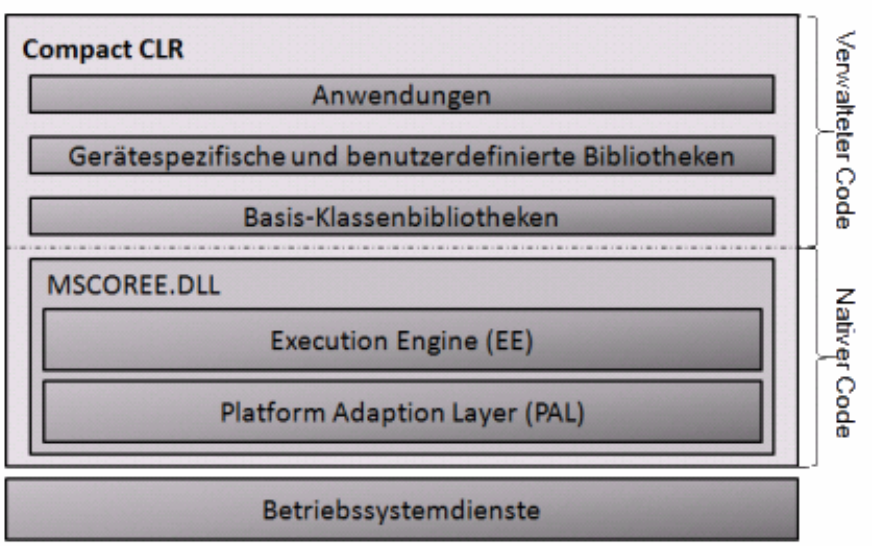

Abbildung 3: Architektur des .NET Compact Framework nach Wigley et al. (2003: 40f.)

Vorteil dieser modularen Architektur ist die Ausführbarkeit von .NET-Anwendungen auf beliebigen Plattformen, für welche eine .NET-Ablaufumgebung zur Verfügung steht. Eine Erweiterung auf eine neue Plattform wird dabei durch den Platform Adaption Layer erleichtert, da dieser den größten Teil der plattformspezifischen Funktionalität kapselt und daher einfach ersetzt werden kann (vgl. ebd.: 41).

[19] Assembly = Ressource, die verwalteten .NET-Code enthält und in der Regel die Dateiendung DLL oder EXE besitzt.

2.2.2.2 Application Domains

Bei einer Application Domain[20] handelt es sich wie bei den Application Domains im .NET-Mutterframework um einen CLR-verwalteten Bereich (vgl. ebd.: 42). Jede .NET-Anwendung wird innerhalb einer Application Domain ausgeführt, wobei wiederum eine Menge von Application Domains von einer CLR verwaltet werden kann. Eine CLR wird wiederum in einem Betriebssystemprozess ausgeführt. Im Vergleich zu Betriebssystemprozessen bieten Application Domains nach ebd. folgende Vorteile:

- Strikte Isolierung von Anwendungen innerhalb einer Application Domain und Verhinderung direkter Objektaufrufe von einer Anwendungsdomäne in eine andere
- Möglichkeit Anwendungen in einer neuen Application Domain innerhalb eines Prozesses zu starten, ohne laufende Anwendungen der selben Application Domain anhalten zu müssen
- Verhinderung von ungültigen Speicherzugriffen und Sicherstellung von Typsicherheit durch einen Pflicht-Verifizierungsprozess für alle auszuführenden Anwendungen und Verhinderung von Folgen eines Absturzes einer Anwendung in einer Application Domain auf andere Application Domains

2.2.2.3 On-Demand-Kompilierung

MSIL-Code wird nach ebd. erst bei der Ausführung auf Methodenbasis in prozessorspezifischen Code kompiliert. Damit wird sichergestellt, dass nur die Methoden in nativer Form vorliegen, welche auch tatsächlich benötigt werden, was zum Einen Speicherplatz einspart und zum Anderen auch die Performance erhöht, da nur der für die aktuelle Programmausführung benötigte Code kompiliert werden muss und bei Bedarf nativ ausgeführt wird.

Um mehrfache Zugriffe auf dieselben Funktionen zu optimieren und Kompilierschritte einzusparen, wird der erzeugte MSIL-Code in einem Cache abgelegt, welcher von der Garbage Collection bzw. dem Garbage Collector (vgl. Kapitel 2.2.2.4) verwaltet wird.

Diese Strategie, die auch als On-Demand-Kompilierung bezeichnet werden kann, spielt besonders bei mobilen Geräten aufgrund der zumeist stark beschränkten Hardware ihre Vorteile aus.

2.2.2.4 Garbage Collector

Der Garbage Collector, der ebenso im regulären .NET-Framework existiert, ist dafür zuständig, im Speicher befindliche Objekte, falls diese nicht mehr verwendet werden, automatisch freizugeben, sodass die von der klassischen Programmierung mit Zeigern bekannten Speicherlecks verhindert werden können (vgl. ebd.: 45).

Wenngleich dieses Konzept in der Desktopapplikationsentwicklung mittlerweile zum Standard avanciert, stellt es im Entwicklungsbereich für mobile Geräte aufgrund der Hardwarerestriktionen besondere Herausforderungen an eine effiziente Implementierung. Details zur

[20] Detailinformationen zu Application Domains im Internet: http://msdn2.microsoft.com/en-us/library/dah4cwez.aspx.

.NET Compact Framework Garbage Collection und den Unterschieden zur .NET Framework-Implementierung werden von Pratschner (2004) beschrieben.

2.2.2.5 Remote Debugging

Debugging ermöglicht die Analyse, Isolierung und Behebung von Programmfehlern durch Funktionen wie schrittweise Ausführung und Auswertung der Programmvariablen und Objekte während der Ausführungsphase. Remote Debugging erweitert diese Definition dahingehend, dass die Anwendung zur Durchführung des Debugging auf einem anderen System als die zu debuggende Anwendung ausgeführt wird. Diese Funktion ist besonders bei mobilen Geräten von größter Bedeutung um die auf mobilen Geräten installierten Anwendungen komfortabel vom Arbeitsplatzrechner des Entwicklers aus zu testen (vgl. Wigley et al., 2003: 46f.).

Die .NET Compact-Implementierung des Debuggers bietet dabei in Version 3.5 nahezu alle Komfortfunktionen, wie sie auch für Desktopanwendungen angeboten werden, wie das Setzen von Haltepunkten und das Drilldown bis hin zur Objektstruktur-Memberwert-Ebene, um aktuelle Werte und Zustände von Objekten und deren Substrukturen zu analysieren.

2.2.2.6 Basis-Klassenbibliotheken

Neben der Laufzeitumgebung wird bereits eine Vielzahl von Klassen ab Werk mit ausgeliefert. Dabei lehnt sich das .NET Compact Framework an das .NET Framework an, wenngleich die Anzahl der bereitgestellten .NET Compact Framework-Klassen nur einem Bruchteil der .NET-Framework-Klassen entspricht.

In Version 1 werden nach Rubin und Yates (2003: 34f.)

- Mscorlib.dll mit den Basisdatentypen sowie Text-, Threading-, Diagnose- und Reflectionfunktionalität,
- System.dll mit regulären Ausdrücken und Ressourcenadressierungsklassen,
- Microsoft.VisualBasic.dll mit Visual Basic-spezifischen Erweiterungen,
- System.Windows.Forms.dll mit GUI-Komponenten,
- System.Windows.DataGrid.dll für das komplexe Datenanzeigesteuerelement DataGrid,
- System.Drawing.dll für Zeichenoperationen,
- Microsoft.WindowsCE.Forms.dll für Windows CE-spezifische Komponenten,
- System.SR.*.dll für Ressourcen-Exceptions,
- System.Data.dll für Datenbankzugriffe und Datenverarbeitung,
- System.XML.dll für XML-Datenverarbeitung,
- System.Web.Services.dll zum Zugriff auf Webdienste,
- sowie System.Net.IrDA.dll zur Kommunikation über den Infrarot-Port standardmäßig ausgeliefert.

Wigley et al. (2003: 57ff.) untergliedern die Basisfunktionalität in

- Base für Datentypen, Collections, IO und Sicherheit,
- Data and XML für ADO.NET-Funktionalität und XML-Datenverarbeitung,
- XML Web Services für die Arbeit mit Webdiensten
- und Windows Forms zur Erstellung graphischer Benutzeroberflächen.

In der aktuellen Version 3.5 des .NET Compact Framework besteht nun neben der in Version 2 erfolgten Erweiterung um einige Komponenten und Standards wie IPv6 eine Unterstützung von LINQ und WCF, wodurch komplexe Datenselektionen und hochflexible Datenanbindungen möglich werden (vgl. Kapitel 2.2.1).

2.2.3 Plattformen

2.2.3.1 Unterstützte Software-Plattformen

Im Rahmen der verschiedenen .NET Compact Framework-Versionen werden nach Microsoft Corporation (o.J.b) folgende Gerätetypen und Plattformen unterstützt:

- **Pocket PC-Betriebssysteme**: Pocket PC 2000, Pocket PC 2002, Windows Mobile 2003 for Pocket PC, Windows Mobile 2003 for Pocket PC SE, Windows Mobile 5.0 for Pocket PC, Windows Mobile 6.0 for Pocket PC
- **Smartphone-Betriebssysteme**: Windows Mobile 2003 for Smartphone, Windows Mobile 5.0 software for Smartphone, Windows Mobile 6.0 for Smartphone
- **Embedded CE-Betriebssysteme**: Windows CE 4.1, Windows CE 4.2, Windows CE 5.0, Windows Embedded CE 6.0

Nach Wilson (2007) sind die Grenzen zwischen den verschiedenen Gerätetypen in der heutigen Zeit als fließend zu bewerten, weshalb eine völlige Separation der Betriebssysteme hinsichtlich der verschiedenen Gerätetypen nicht mehr zeitgemäß ist. Ein Resultat dieser Feststellung ist die aktuelle Version 6, genauer 6.1, von Windows Mobile, bei der die Unterscheidung nach Anwendungen bzw. Leistungsklassen erfolgt. Varianten dieser auf Windows CE 5.2-basierenden Plattform werden dabei für die verschiedenen Gerätetypen eingesetzt, sodass sich folgende Zuordnung im Vergleich zu den jeweils vorherigen Versionen[21] ergibt (vgl. ebd.):

- Windows Mobile 6 Standard als Nachfolger von Windows Mobile 5.0 für Smartphone
- Windows Mobile 6 Professional als Nachfolger von Windows Mobile 5.0 für Pocket PC Phone Edition
- Windows Mobile 6 Classic als Nachfolger von Windows Mobile 5.0 für Pocket PC

Trotz der verschiedenen Varianten kann der Entwickler nach ebd. stets auf die gleiche Basis-API und die zugehörigen Tools bei der Entwicklung zurückgreifen.

2.2.3.2 Unterstützte Prozessorarchitekturen

Nach Wilson (2005b) werden im Rahmen von .NET Compact Framework der Version 2.0 unter Anderem folgende Prozessorarchitekturen unterstützt:

- ARM
- SH4
- MIPS
- X86

[21] Eine Übersicht über vergangene, aktuelle und geplante Versionen von Windows Mobile ist unter http://www.wolfgang-rolke.de/wince/platform.htm verfügbar.

Allgemein ist das .NET Compact Framework auf allen Prozessorarchitekturen lauffähig, die von den unter Kapitel 2.2.3.1 angeführten Betriebssystemen unterstützt werden.

2.2.4 Datenanbindung

Das .NET Compact Framework besitzt bereits standardmäßig eine Vielzahl an Kommunikationsschnittstellen. Dies schließt die Bereiche

- Datenbankanbindung,
- Dateisystemressourcen,
- verteilte, entfernte Dienste, z.B. Webdienste,
- Infrarotkonnektivität
- und klassische Socketkommunikation mit ein.

Im Folgenden sollen nun die bedeutendsten Funktionalitäten im Bereich Datenanbindung betrachtet werden.

2.2.4.1 Datenbankunterstützung

Das .NET Compact Framework unterstützt standardmäßig Microsoft SQL Server und dessen mobile Varianten, wie den SQL Server CE bzw. dessen Nachfolgeversion SQL Server Mobile, welcher nun als SQL Server Compact bekannt ist (vgl. Microsoft Corporation, o.J.c; Microsoft Corporation, o.J.d). Im Unterschied zu der SQL Server-Desktopversion können die mobilen SQL Server-Versionen jedoch direkt auf dem mobilen Endgerät installiert werden und als lokaler Datenspeicher dienen.

Nach Wildermuth (2005) kann neben dem SQL Server von Microsoft auch Sybase iAnywhere[22] oder Pocket Access[23] zur Datenpersistierung eingesetzt werden. Hierzu sind aber entsprechende Treiber bzw. Bibliotheken mit der zu erstellenden Anwendung auf das mobile Endgerät auszuliefern.

Generell gibt es für nahezu alle bekannten Datenbankmanagementsysteme .NET Compact Framework-Bibliotheken zum Ansprechen dieser DBMS. Unter Anderem werden Oracle, MySQL, PostgreSQL, DB2, Interbase und Firebird unterstützt (vgl. ComponentSource, o.J.).

Aufgrund der generischen Abstract-Factory-Pattern-Architektur (vgl. Gamma et al., 1995) von ADO.NET (vgl. Mosa, 2006) können weitere Provider bei Bedarf entwickelt und unmittelbar benutzt werden, ohne dass eine Anpassung des .NET Compact Framework erforderlich wird.

Abgefragte Daten können programmatikalisch über DataSets (vgl. Pohmann, 2004) im Speicher gehalten, gefiltert und manipuliert werden. Seit Version 3.5 besteht nach Hart (2007) zudem eingeschränkte LINQ-Unterstützung, sodass komplexe Abfragen auf Datasets unter Zuhilfenahme von Lambda-Ausdrücken (vgl. Troelsen, 2007: 374-381) deklarativ und kompakt formuliert werden können. Primäre Einschränkungen sind nach Hart (2007) in der .NET Compact Framework-Implementierung, dass nur Standard-Operatoren unterstützt

[22] Sybase iAnywhere ist ein DBMS von Sybase; im Internet: http://www.sybase.com/ianywhere.
[23] Bei Pocket Access handelt es sich um eine kompakte Version von Microsoft Access; vgl. Roof (2003).

werden und die Provider LINQ to SQL und LINQ to Entities aus Speicherplatzgründen nicht unterstützt werden.

LINQ besteht dabei nach Klein (2008: 3) grundsätzlich aus Standard-Abfrageoperatoren, die die zugrunde liegende Architektur für Filter-, Suchen-, Sortieren- und Ausführungs-Aktionen auf Basis nahezu beliebiger Datenquellen, wie z.B. XML[24], relationalen Daten[25], ADO.NET DataSets[26] oder im Speicher befindliche Objektstrukturen[27], bilden. Nach Hossain (2007) besteht zudem die Möglichkeit, benutzerdefinierte LINQ-Provider für beliebige Datenquellen zu erstellen.

Besonders im Umfeld der Anbindung von Datenbanken zeigt sich nach Klein (2008: 8) im Rahmen der Unterstützung streng typisierter Datenobjekte ein bedeutender Vorteil von LINQ, da keine unsicheren Datentypumwandlungen, welche man als Casts bezeichnet, mehr erforderlich sind.

Die Standard-IDE von Microsoft, Visual Studio (vgl. Kapitel 2.2.5), bietet ferner eine LINQ-Integration mit Assistentenunterstützung, z. B. über den Object Relational Designer, welche die graphische, assistentengestützte Erstellung von LINQ-to-SQL-Entitätsklassen ermöglicht (vgl. Klein, 2008: 287-310).

Ein Beispiel für eine einfache LINQ-Abfrage wird unter Listing 1 vorgestellt. Diese ermittelt aus einer Liste alle Studentinnen, die jünger als 25 Jahre sind und gibt diese mit Name, Vorname und Alter aufsteigend sortiert nach dem Namen aus. Herkömmliche Lösungen würden hier eine Realisierung mit Schleifenkonstrukten und Kontrollstrukturen erfordern.

Als großer Vorteil ist zudem die IntelliSense[28]-Unterstützung bei der Eingabe der LINQ-Anweisungen hervorzuheben, da aufgrund der im Vergleich zu herkömmlichen SQL strengen Typisierung die jeweiligen Membereigenschaften[29] über Reflection ausgelesen werden können um dem Anwender Autovervollständigung zu bieten.

```
enum Geschlecht { Maennlich, Weiblich }
class Student
{
    public string Name { get; set; }
    public Geschlecht Geschlecht { get; set; }
    public int Alter { get; set; }
}

// studs sei vom Typ List<Student> und enthalte eine Menge von Studenten
var ergebnisse = from p in studs
      where p.Geschlecht == Geschlecht.Weiblich && p.Alter < 25
      orderby p.Name select new { p.Name, p.Vorname, p.Alter };
foreach(var ergebnis in ergebnisse) Console.WriteLine(ergebnis);
```

Listing 1: Exemplarische In-Memory-LINQ-Abfrage in C#

[24] Realisierung durch LINQ-to-XML-Provider.
[25] Implementierung durch LINQ-to-SQL-Provider.
[26] Umsetzung durch LINQ-to-DataSet-Provider.
[27] Realisierung durch LINQ-to-Objects-Provider.
[28] Technologie von Microsoft zur automatischen Code-Vervollständigung in Visual Studio.
[29] Instanzspezifische Eigenschaften einer Klasse und damit Eigenschaften eines konkreten Objekts.

2.2.4.2 Verteilte, entfernte Dienste

Von Beginn an bot das .NET Compact Framework eine Unterstützung zur Konsumierung von XML-Webdiensten an (vgl. Wigley et al., 2003: 511-536).

Mit Version 3.5 des .NET Compact Framework führt Microsoft die Unterstützung von WCF, der Windows Communication Foundation, ein. Im Gegensatz zur vollständigen WCF-Implementierung des .NET Framework 3.5 unterliegt die .NET CF-Realisierung jedoch einigen Einschränkungen, wie beispielsweise der fehlenden TCP-Binding-Unterstützung oder der Nicht-Verfügbarkeit des binären Formatters (vgl. Arnott, 2007).

Bei WCF handelt es sich nach Microsoft Corporation (2007) um ein dienstorientiertes Programmiermodell, welches auch als WCF Service Model bezeichnet wird. WCF ermöglicht damit, ähnlich wie Webservices, die Kommunikation zwischen verschiedenen Diensten, wenngleich WCF die konkrete Kommunikationstechnologie abstrahiert. Ein großer Vorteil dieser Technologie liegt dabei darin, dass Entwickler ein einheitliches Programmiermodell für eine Vielzahl von Kommunikationstechnologien wie ASP .NET Webservices, .NET Framework Remoting, Enterprise Services, WSE und MSMQ erhalten (vgl. ebd.).

Funktionalität, die im Sinne einer serviceorientierten Architektur (SOA) für andere Anwendungen oder Dienste bereitgestellt werden soll, wird bei WCF anhand von Contracts (vgl. Löwy, 2007: 7-11) beschrieben, wobei es sich bei einem Contract um eine in einer .NET-Programmiersprache vorliegende Schnittstellendefinition, die die Funktionalität des jeweiligen Dienstes beschreibt, handelt.

Wie in Abbildung 4 visualisiert, werden die zu bereitstellenden Dienste als WCF Service innerhalb eines Host-Prozesses bereitgestellt. Die Schnittstellenbeschreibung in Form des Contracts ist ebenso wie die Implementierung völlig unabhängig von der Technologie, über welche die jeweiligen Dienste angeboten werden. So kann neben der programmatischen Variante auch rein über eine Konfigurationsdatei festgelegt werden, in welcher Form ein Dienst interessierten Konsumenten, den WCF Clients, zur Verfügung gestellt werden soll.

Neben einem Kontrakt zur Beschreibung der Dienstfunktionalität erweist sich auch das Binding als wichtiger Aspekt, da es das für die Kommunikation zu verwendende Protokoll spezifiziert (vgl. ebd.: 18-21)

Als letzte zentrale Begrifflichkeit ist schließlich noch die Eigenschaft Address zu nennen, welche die Lokation der Dienstbereitstellung beschreibt (vgl. ebd.: 4ff.).

Eine Kombination aus Contract, Binding und Address wird im Rahmen von WCF als Endpoint oder Endpunkt bezeichnet[30]. Ein WCF Service kann dabei beliebig viele Endpunkte besitzen und somit beliebig viele Clients auf Basis beliebig vieler, unterschiedlicher Transportprotokolle bedienen.

Im in Abbildung 4 dargestellten Beispiel stellt der WCF-DemoService-Dienst seine Funktionalität sowohl über HTTP, als auch über TCP zur Verfügung. Beachtet werden sollte hierbei, dass die Implementierung der Logik völlig unabhängig von den bei der Kommunikation verwendeten Protokollen ist. Zudem kann nicht nur das Transportprotokoll, sondern auch die Kodierung der Daten, z.B. XML oder binär, pro Endpunkt über das Binding-

[30] Man spricht hier auch vom ABC-Prinzip (Address, Binding, Contract). Vgl. Löwy (2007).

Attribut festgelegt werden. Auch Aspekte wie Sicherheit oder Transaktionalität sind über WCF konfigurierbar (vgl. Microsoft Corporation, 2007).

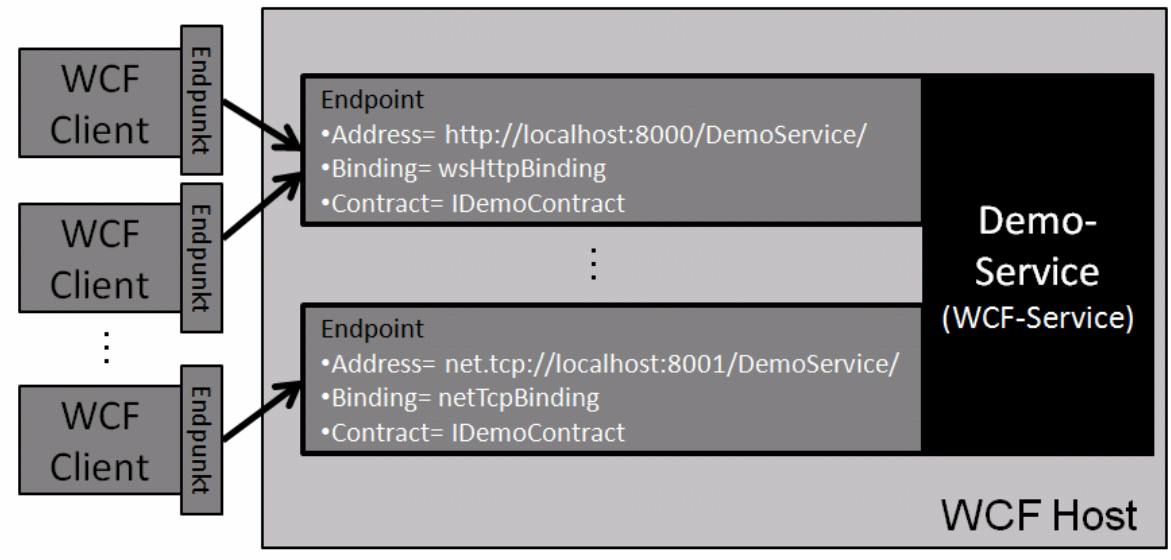

Abbildung 4: Exemplarische WCF-Kommunikationsstruktur

2.2.5 Entwicklungsumgebungen

Die Entwicklung von .NET Compact Framework-Anwendungen erfolgt aufgrund der Hardwarebeschränkungen von mobilen Geräten in der Regel auf einer Workstation, da dort auch entsprechende komfortable Entwicklungswerkzeuge bereitstehen.

Wenngleich freie Entwicklungsumgebungen wie SharpDevelop[31] .NET Compact Framework-Unterstützung bieten (vgl. IC#Code, o.J.), handelt es sich bei Visual Studio von Microsoft um die Standard-IDE[32] zur Entwicklung von mobilen Anwendungen basierend auf der .NET Compact Framework-Plattform. Dies liegt zum Einen darin begründet, dass Visual Studio das Standardprodukt von Microsoft für die Entwicklung von .NET-basierten Anwendungen darstellt und zum Anderen nach Wigley et al. (2003: 18f.) die identischen und den Entwicklern daher in der Regel bereits vertrauten Tools und Technologien wie bei der Desktopentwicklung benutzt werden können

Darüberhinaus besteht nach ebd. die Möglichkeit, die Anwendung zu Testzwecken auf das Effektivgerät oder in eine virtuelle Maschine zur Emulation zu deployen, das Debugging jedoch über die integrierte Remote-Debugging-Funktionalität von Visual Studio abzuwickeln, sodass das Testen der Anwendung auf der jeweiligen Workstation unter Zuhilfenahme der umfangreichen Debugging-Funktionalitäten von Visual Studio erfolgen kann.

Um Anwendungen für mobile Endgeräte auch ohne dieselben auszuführen und zu testen, kann wie bereits angesprochen, ein Device Emulator eingesetzt werden (vgl. Abbildung 5).

[31] Im Internet: http://www.icsharpcode.net/OpenSource/SD/.
[32] IDE = Integrated Development Environment; integrierte Entwicklungsumgebung.

Abbildung 5: Geräteemulator im Einsatz

Mit der aktuellen Version 3.5 des .NET Compact Framework und Visual Studio 2008 werden bereits mehrere Emulatoren ab Werk bereitgestellt. Dabei hat man die Auswahl zwischen verschiedenen mobilen Endgeräten und Sprachversionen, wobei eine bestimmte Konfiguration als Image auswählbar ist (vgl. Abbildung 6). Zusätzliche, z.B. lokalisierte, Images sind über Microsofts Internetpräsenz erhältlich[33]. Nach Feinman (2006) verfügt bereits der mit Visual Studio 2005 bereitgestellte Geräteemulator über die meisten Features des physikalisch echten Gerätes. Dies schließt nach ebd. selbst die Simulation von Speicherkarten und die Unterstützung von DMA[34]-Zugriffen mit ein.

Im Rahmen der neuen Produktversion 2008 von Visual Studio werden nach Chopra (2007) die integrierte und durch die IDE unterstützte Erstellung von Unit Tests[35] für mobile Anwendungen sowie der Device Security Manager zum Testen der erstellten Anwendungen unter Berücksichtigung von auf mobilen Endgeräten vorfindbaren Security Policies eingeführt.

Neben Verbesserungen in der Ausführungsgeschwindigkeit sind nach Chopra (2007) vor allem die erweiterte Hardware-Emulation von Headset, Mikrofon und Batterie zu nennen, wodurch eine relativ realitätsgetreue Simulation der erstellten Anwendungen und damit auch ein Testen der Applikationen bei verschiedenen Umgebungsparametern möglich wird.

[33] Beispielsweise sind lokalisierte Images der aktuellen Windows Mobile 6.1-Version unter http://www.microsoft.com/downloads/details.aspx?FamilyID=3d6f581e-c093-4b15-ab0c-a2ce5bffdb47&DisplayLang=en verfügbar.
[34] DMA = Direct Memory Access; ermöglicht Peripheriegeräten einen direkten Zugriff auf den Arbeitsspeicher ohne Erfordernis der Involvierung des Prozessors.
[35] Unit Tests = Tests von Softwarekomponenten zur Verifizierung der einwandfreien Funktionalität; Unit Tests können sich im Rahmen der objektorientierten Programmierung auf Methoden, Klassen oder andere Entitäten mit Komponentencharakter beziehen.

Abbildung 6: Geräteemulator-Manager

2.2.6 Ausblick

Zu dem Nachfolger des .NET Compact Framework 3.5 sind derzeit noch keine Details bekannt. Die zugrunde liegende Betriebssystemplattform Windows Mobile soll jedoch nach aktuellem Stand zusätzlich zu der mit Version 6.1 bzw. Internet Explorer Mobile 6 eingeführten Unterstützung von H.264, der Unterstützung von Gruppenrichtlinien für mobile Geräte sowie der vereinfachten Netzwerkverbindungskonfiguration (vgl. Widmann, 2008) in Version 7 nach Weinberg (2008) unter Anderem folgende neuen Funktionen bieten:

- Umfassende Motion Gestures-Unterstützung:
  - Herkömmliche, fingerbezogene Motion Gestures für Bildlaufleisten, Task-Menü, Shortcuts, Listensuche
  - Aktionen über Bewegung bzw. Lage des Gerätes
  - Anpassung der Darstellung (horizontale und vertikale Bilddarstellung)
  - Schütteln des Mobilteils für Slideshow-Navigation und Musiknavigation
- Aktionen basierend auf der Lichtintensität, z.B.:
  - Mobilgerät schaltet Display automatisch aus und auf Vibrationsalarm um, wenn es in Tasche (verringerte Lichtintensität) gelegt wird
  - Automatische Verbindung mit Remote-Geräten bei Blickkontakt mit Kamera
  - Stummschaltung des Gerätes, wenn Hand über Kamera gehalten wird
- Handschriftenerkennung für OneNote Mobile

2.3 Konklusion

Beim .NET Compact Framework handelt es sich um ein umfangreiches und flexibles Framework zur Realisierung von Applikationen mit dem Anwendungsbereich mobile Endgeräte und eingebettete Systeme. Die Laufzeitumgebung sowie die Programmierparadigmen ähneln weitgehend der durch das .NET Framework geprägten Desktopentwicklung.

Hinsichtlich der Plattformen werden Geräte auf Basis von ARM-, SH4-, MIPS- und X86–Prozessoren sowie grundsätzlich alle Architekturen, auf denen Windows Mobile-Betriebssysteme ablauffähig sind, unterstützt, womit die meisten marktrelevanten Plattformen abgedeckt werden.

Im Bereich Datenbankmanagementsysteme werden nahezu alle bekannten Systeme unterstützt. Über Technologien wie LINQ besteht zudem eine komfortable und mächtige Möglichkeit, Objektstrukturen zu durchsuchen. Durch die Unterstützung von WCF ist die Anbindung prinzipiell beliebiger lokaler und entfernter Dienste bei Beibehaltung maximaler Flexibilität und eines einheitlichen Entwicklungsmodells möglich. Lediglich die aus Speicher- und Performancegründen vorliegenden Einschränkungen der .NET CF-Implementierung im Hinblick auf WCF und LINQ schränken die Mächtigkeit und Flexibilität der genannten Technologien auf mobilen Endgeräten noch ein.

Die Entwicklung auf Basis des nun in Version 3.5 vorliegenden .NET Compact Framework kann neben freien Entwicklungsumgebungen über Visual Studio erfolgen, wodurch Komfortfunktionen wie IntelliSense, Debugging und ein Geräteemulator zur Verfügung stehen.

Im Folgenden sollen nun auf Basis der vorgestellten Technologien entwickelte Anwendungen betrachtet werden.

3 Anwendungen

Für Microsofts .NET Compact Framework-Plattform existiert eine Vielzahl von Anwendungen in den verschiedensten Anwendungsgebieten. Dieses Kapitel soll einen kurzen Überblick über die Vielfalt dieser Anwendungen außerhalb der Standardanwendungsgebiete Office-Applikationen[36] und PIM[37]-Systeme geben, indem thematisch voneinander abgegrenzte Applikationen vorgestellt werden, wenngleich die Übersicht aufgrund des vorgegebenen, begrenzten Rahmens dieser Arbeit nur Ausschnittscharakter aufweisen kann. Obwohl in diesem Kapitel auch Anwendungen mit partiell ortsabhängigem Charakter aufgezeigt werden, soll hinsichtlich von Anwendungen mit Zentrierung auf den Aspekt LBS auf Kapitel 4.4 verwiesen werden.

Eine umfassende, aber unvollständige Übersicht über frei verfügbare .NET Compact Anwendungen wird auf Microsofts Open Source Plattform CodePlex[38] gegeben. Eine relativ aktuelle Liste aller am Markt erhältlichen, auch kommerziellen, .NET Compact Framework-Anwendungen ist unter dem Blog des .NET Compact Framework-Teams[39] einsehbar.

3.1 eAssessments

Nach Microsoft Corporation (2003) handelt es sich bei eAssessments um eine beim Royal Melbourne Hospital eingesetzte, auf .NET-Framework- und .NET-Compact-Framework-

[36] Bekannte Office-Anwendungen im Umfeld der .NET-Compact-Framework-Betriebssystemplattform Windows Mobile sind Pocket Word, Pocket Excel und Pocket PowerPoint.
[37] PIM = Personal Information Manager; Software zur Verwaltung von E-Mails, Kontakten, Terminen, Aufgaben und ähnlichen Entitäten; Outlook Mobile ist z.B. ein PIM-System für Windows Mobile-Geräte.
[38] Im Internet: http://www.codeplex.com/.
[39] Im Internet: http://blogs.msdn.com/netcfteam/archive/2005/06/23/431948.aspx.

Technologie basierende Anwendung zur Verbesserung des Patientenbegutachtungsprozesses.

Die bisher auf Papier und in Dateiablagen erfolgte Dokumentation von Untersuchungen hatte nach ebd. zeitaufwändige und monotone Suchvorgänge und Analysen zur Folge.

eAssessments besteht aus mehreren Anwendungen für die Bereiche Konsultation und Administration. Die Konsultationssoftware befindet sich nach ebd. komplett auf dem jeweiligen mobilen Endgerät und wird als Unterstützung für Patientenuntersuchungen benutzt. Jede Abteilung besitzt spezifische Anforderungen, weshalb spezielle Applikationen für die verschiedenen Abteilungen zum Einsatz kommen. Informationen werden lokal in einer SQL Server CE-Datenbank gespeichert und mit einem zentralen SQL Server synchronisiert.

3.2 Beast Master.Net

Beast Master.NET[40] ist eine kommerzielle .NET Compact Framework-Anwendung von Kai Bruchmann mit dem Einsatzzweck der Ungezieferabwehr auf Basis von Hochfrequenzen.

Beast Master .NET unterstützt dabei nach Bruchmann (2007) für verschiedene Tiere, wie Stechmücken, Ratten und Kakerlaken, vorkonfigurierte Frequenzprofile zwischen 7 kHz und 12 kHz sowie 16 kHz und 20 kHz.

Die Wirksamkeit von auf Hochfrequenz beruhenden, tiervertreibenden Technologien ist jedoch nicht bewiesen.

3.3 USPS Intelligent Mail Barcode (OneCode)

Der USPS OneCode 4-State Customer Barcode[41] soll aufgrund einer effizienteren, auf vier Balken beruhenden Kodierung wesentlich mehr Informationen auf gleichem Raum unterbringen als bisherige Barcode-Ansätze.

USPS[42] Intelligent Mail Barcode (OneCode) implementiert hierbei einen Encoder für dieses Barcode-System, der auf mobilen Endgeräten ablauffähig ist.

3.4 NXT.NET

Nach György (2008) handelt es sich bei NXT.NET[43] um eine in C# 3.0 entwickelte Bibliothek für das .NET Framework 3.5 und das .NET Compact Framework 3.5, die es nutzenden Anwendungen erlaubt, LEGO Mindstorms NXT[44]-Systeme zu steuern.

[40] Im Internet: http://www.pocketkai.net/asp/details.aspx?249.
[41] Im Internet: http://www.codeplex.com/onecode; vgl. http://ribbs.usps.gov/onecodesolution/.
[42] USPS = United States Postal Service, US-amerikanischer Postdienstleister.
[43] Im Internet: http://www.codeplex.com/NxtNet.
[44] Produktserie von LEGO, um Roboter und andere autonome sowie interaktive Systeme zu entwickeln. Im Internet: http://mindstorms.lego.com/.

NXT.NET wird nach ebd. vom ungarischen MSDN Competence Center entwickelt und enthält neben der Mobile-Bibliothek NxtNet.MobileLib eine mobile Anwendung namens NxtNet.MobileApp zum Testen der Bibliotheksfunktionen und zum Steuern eines LEGO Mindstorms NXT-Roboters.

3.5 Konklusion

Es wurde gezeigt, dass die Anwendungsmöglichkeiten des .NET Compact Framework weit gestreut sind. Neben Standard-Anwendungen, wie z. B. Office-Applikationen existieren Anwendungsbereiche im klinischen Umfeld, in der Logistik, im Spielebereich und selbst in der Ungezieferbekämpfung.

Anwendungen ortsabhängiger Dienste sollen nun ebenso wie die Unterstützung von Location Based Services auf Betriebssystem-, Framework- und Datenbankebene im folgenden Kapitel aufgezeigt werden.

4 Unterstützung ortsabhängiger Dienste

In diesem Kapitel sollen die Spezifika der .NET Compact Framework-Technologie und häufig benutzter Softwarekomponenten im Kontext dieser Plattform im Hinblick auf den Aspekt ortsabhängige Dienste untersucht werden.

4.1 Auf Betriebssystemebene

Für die Betrachtung der Unterstützung ortsabhängiger Dienste soll Windows Mobile in der aktuellen Version 6 zugrunde gelegt werden, da dieses die aktuelle Software-Plattform für .NET Compact Framework-Anwendungen darstellt.

Neben der grundsätzlichen Unterstützung von Kommunikations- und Ortsbestimmungstechnologien bietet Windows Mobile einige Emulationsfunktionalitäten, um den Entwicklungsprozess ortsabhängiger Anwendungen zu erleichtern.

4.1.1 Emulationsfunktionalität

Nach Wilson (2007) nutzen Mobilfunkgeräte naturgemäß primär Batteriestrom. Abhängig vom Aufenthaltsort des jeweiligen Mobilfunkgerätes kann die Option auf Wechselstrom bestehen oder fehlen. Mobile Anwendungen müssen daher nach ebd. bei Energieabfall mit definiertem Verhalten reagieren und Datenverlust verhindern. Um die Einflussgröße Ladezustand bereits bei der Entwicklung zu berücksichtigen, wird wie bereits unter Kapitel 2.2.5 angesprochen, die Emulation des Batterieladezustands unterstützt.

Ferner kann nach ebd. auch Kommunikations- und Telefonieverhalten simuliert werden. Dies schließt die Simulation der Durchführung ein- und ausgehender Telefonate und SMS mit ein. Zusätzlich können auch Situationen simuliert werden, bei denen aufgrund schlechten Empfangs Verbindungsabbrüche auftreten. Daneben sind nach ebd. auch Netzwerk-

handoffs[45] zwischen 2G[46]- und 3G[47]-Netzwerken komplett ohne physikalisch vorhandenes Mobilgerät simulierbar.

4.1.2 FakeGPS

Besondere Bedeutung im Kontext ortsabhängiger Dienste besitzt das Windows Mobile-Dienstprogramm FakeGPS. Dieses ermöglicht nach ebd. den Test standortaktivierter Anwendungen ohne das Vorhandensein eines physikalischen GPS-Geräts. FakeGPS tritt dabei nach außen als ein GPS-Gerät auf, dessen GPS-Informationen jedoch aus einer vom Entwickler spezifizierbaren Textdatei mit GPS NMEA-Nachrichten[48] entstammen. Die vereinfachte Erstellung der entsprechenden GPS NMEA-Nachrichten kann dabei mit Hilfe des Tools Fake GPS Helper[49] erfolgen.

Abschließend soll noch das Local Server Framework Erwähnung finden, welches das Testen von mobilen Diensten mit Zugriff auf Remote-Dienste gestattet (vgl. ebd.).

4.2 Auf Frameworkebene

Auf Frameworkebene liegt sowohl eine Unterstützung durch das .NET CF selbst, als auch über darauf basierende Fremdbibliotheken vor. Eine verwaltete API zum Zugriff auf Ortsbestimmungsinformationen ist in der aktuellen Version des .NET CF noch nicht erhalten. Jedoch besteht die Möglichkeit über native API-Zugriffe beispielsweise die aktuelle Mobilfunkzelle, in der sich der Mobilfunkgerätanwender befindet, auszulesen.

Daneben existieren auf dem .NET CF basierende verwaltete Fremdbibliotheken, was im folgenden näher betrachtet wird.

4.2.1 SharpGPS

Ein wesentliches Erfordernis für die Realisierung ortsabhängiger Dienste ist die Möglichkeit des verwendeten Entwicklungsframeworks mit der GPS-Geräteeinheit zu kommunizieren.

Neben der reinen SDK-Unterstützung unter Windows Mobile 5 besteht nach Nielsen (2007a) über die frei erhältliche, über CodePlex verfügbare, Bibliothek SharpGPS eine plattformübergreifende Möglichkeit zur Kommunikation mit einer GPS-Einheit unter .NET Framework 2.0 und .NET Compact Framework 2.0. SharpGPS unterstützt die wichtigsten NMEA-Datensatzarten[50]. Im Detail handelt es sich nach Nielsen (2007b) um folgende NMEA-Datensatzarten:

- GPGGA - Global Positioning System Fix Data

[45] Handoff = Übertragung eines eingehenden Anrufs bzw. einer eingehenden Datensitzung von einem Netzwerk zu einem anderen.
[46] Zweite Generation der Mobilfunkkommunikation mit Übertragungsraten bis in der Regel 64 kbit/s; hierzu zählt u.a. GSM.
[47] Dritte Generation der Mobilfunkkommunikation mit Übertragungsraten zwischen 64 kbit/s und 384 kbit/s; hierzu zählt u.a. UMTS.
[48] GPS-Daten im Format gemäß einer Spezifikation der National Marine Electronics Association.
[49] Im Internet: http://www.codeplex.com/fakegpshelper.
[50] Liste aller NMEA 0183-Datensätze im Internet: http://www.nmea.de/nmea0183datensaetze.html.

- GPGLL - Geographic position, Latitude and Longitude
- GPGSA - GPS DOP and active satellites
- GPGSV - Satellites in view
- GPRMC - Recommended minimum specific GPS/Transit data
- PGRME - Estimated Position Error (Garmin proprietary)

Damit können aktuelle Informationen zu den Satelliten selbst oder aber auch zur Position des Empfängers abgerufen werden. Auch Detailinformationen wie Antennenhöhe, aktive Satelliten oder empfohlene Übertragungsraten können komfortabel abgerufen werden. Das ausgelieferte Beispielprojekt enthält zudem bereits fertige Anwendungen die Informationen zu den Satelliten und zur aktuellen GPS-Empfängerposition graphisch visualisieren.

4.2.2 Geoinformationssystem- und Berechnungsframeworks

Ferner existiert eine Reihe verschiedener Bibliotheken wie GeoAPI.NET[51] oder Proj.NET[52] um Umwandlungen bzw. Berechnungen in und zwischen geodätischen Koordinatensystemen zu ermöglichen. Zudem fungieren diese Bibliotheken als Basis für eine Vielzahl weiterer ortsabhängiger Bibliotheken und Dienste.

Schließlich soll noch das LinqToGeo-Projekt erwähnt werden, welches das Abfragen und Filtern von Geoinformationssystemdaten (GIS-Daten) auf Basis von LINQ-Technologie (vgl. Kapitel 2.2.4.1) als Zielsetzung besitzt. In diesem Kontext soll auch SharpMap[53] angesprochen werden, wobei es sich dabei um eine Mapping-Bibliothek und Rendering-Engine für Landkarten, mit der Möglichkeit auf GIS-Daten zuzugreifen und diese abzufragen, handelt.

4.2.3 Einbindung bestehender Dienste

Durch die unter Kapitel 2.2.4.2 angesprochene WCF-Unterstützung können auch über eine Internetverbindung verfügbare Kartendienste, wie beispielsweise Google Maps oder Virtual Earth (vgl. Kapitel 4.4), über Webdienste programmatisch angebunden werden.

4.3 Auf Datenpersistierungsebene

4.3.1 SQL Server 2008

Wenngleich eine Vielzahl von Datenbanken für die Benutzung mit .NET Compact Framework in Frage kommt (vgl. Kapitel 2.2.4.1), soll im Rahmen dieser Untersuchung der Fokus auf die von Microsoft angebotene Datenbank-Lösung Microsoft SQL Server in der nach Fulton (2008) für das dritte Quartal 2008 erwarteten Version 2008 gelegt werden, da zwischen den verschiedenen Microsoft-Produkten eine optimale Interoperabilität zu erwarten ist.

[51] Im Internet: http://www.codeplex.com/GeoAPI.
[52] Im Internet: http://www.codeplex.com/ProjNET.
[53] Im Internet: http://www.codeplex.com/SharpMap.

Alternativ wäre hier eine Betrachtung des DBMS SQL Server Compact 3.5 möglich, welches im Gegensatz zu SQL Server 2008 auch direkt auf dem mobilen Endgerät betrieben werden kann. Aufgrund der allgemeinen Tendenz Funktionalität von der leistungsfähigeren Desktopvariante in die mobile Version zu übernehmen, aber der bei größerem Rechenbedarf ohnehin erforderlichen Auslagerung auf ein Serversystem soll jedoch im Rahmen dieser Arbeit die im dritten Quartal 2008 erwartete Version 2008 des Microsoft SQL Server näher beleuchtet werden.

SQL Server bietet nach Aschenbrenner (2008) in der Version 2008 explizit Unterstützung für die Persistierung geometrischer und geographischer Daten durch die Einführung der Spatial Types, wobei es sich frei übersetzt um räumliche Datentypen handelt.

Mit dem Datentypen GEOMETRY können nach ebd. geometrische Formen in einem dreidimensionalen Raum abgebildet werden, wobei folgende konkrete Formen unterstützt werden:

- Point
- MultiPoint
- LineString
- MultiLineString
- Polygon
- MultiPolygon
- GeometryCollection

Da über diesen Datentyp reine geometrische Formen ohne direkte Berücksichtigung externer Umgebungsparameter, wie z.B. die Erdkrümmung, realisiert werden können, spricht man hier nach ebd. auch von Flat-Earth-Daten.

Der Datentyp GEOGRAPHY zielt im Gegensatz dazu auf GPS[54]-Koordinaten ab und erlaubt deren Persistierung. Darüber lassen sich nach ebd. geometrische Formen unter Berücksichtigung von Umgebungsparametern wie der Erdkrümmung abbilden. Man spricht dabei von Round-Earth-Daten.

4.3.2 Sync Services for ADO.NET

Abschließend soll noch auf die Sync Services for ADO.NET, die als Teil des Microsoft Sync Framework[55] (MSF), erhältlich sind, verwiesen werden. MSF löst nach Microsoft Corporation (o.J.e) das Problem, beliebige Daten in einem beliebigen Datenspeicher über ein beliebiges Protokoll und eine beliebige Netzwerktopologie synchron zu halten.

Abbildung 7 visualisiert die bei einem einfachen, exemplarischen Synchronisationsszenario beteiligten Akteure.

[54] GPS = Global Positioning System; satellitengestütztes, weltweites Positionierungssystem.
[55] Im Internet: http://msdn.microsoft.com/en-us/sync/default.aspx.

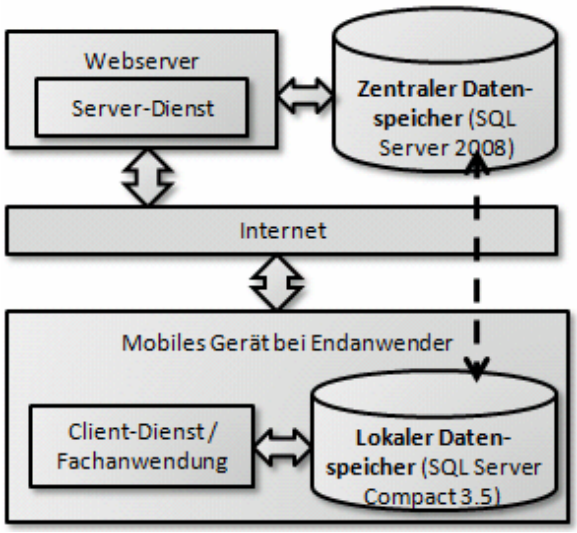

Abbildung 7: Einfaches exemplarisches Synchronisationsszenario

In diesem Szenario greift ein am mobilen Endgerät (Client) installiertes Fachverfahren bzw. ein Dienst über das Internet auf Funktionen zurück, die von dem Server-Dienst eines Webservers angeboten werden. Die originären Daten stammen dabei von einem für alle Benutzer der Serveranwendung und damit für alle mobilen Anwender zentralen Datenspeicher, welcher durch SQL Server 2008 realisiert wird. Der Client besitzt jedoch zusätzlich einen lokalen Datenspeicher in Form einer Installation von SQL Server Compact 3.5 auf dem jeweiligen Endgerät um die vom zentralen Datenspeicher stammenden Daten lokal vorzuhalten und Caching-Effekte oder auch eine Verfügbarkeit an Orten mit schwacher oder fehlender Netzempfangsmöglichkeit zu erzielen. Der gestrichelte Pfeil in Abbildung 7 soll dabei die nicht-physikalische, logische Verbindung zwischen den zu synchronisierenden Datenbeständen aufzeigen.

Sync Services for ADO .NET ermöglicht nach ebd. die Synchronisation zwischen über ADO.NET unterstützten Datenbanken, wenngleich auch ein Austausch mit allen anderen durch MSF unterstützten Datenquellen wie Webdiensten oder benutzerdefinierten Datenquellen möglich ist.

Diese Funktionalität könnte beispielsweise, wie bereits in Abbildung 7 angedeutet, dazu eingesetzt werden, an Orten ohne Internetanbindung Daten über ein mobiles Endgerät manuell zu erfassen und eine automatische Synchronisation bei Erreichen eines Ortes mit Internetanbindung anzustoßen. Umgekehrt könnten Daten an Orten mit Netzverbindung automatisch aus dem Internet bezogen werden, um diese für die lokale Verwendung an Orten mit schlechten Empfangsmöglichkeiten zwischenzuspeichern.

4.4 Auf Anwendungsebene

4.4.1 Virtual Earth Mobile

Aschenbrenner (2008) erläutert, dass eine Visualisierung von GPS-Koordinaten durch Dienste wie Virtual Earth[56] komfortabel und applikationsintegrativ erfolgen kann. Bei Virtual Earth handelt es sich dabei nach ebd. um eine in die jeweilige eigene Anwendung in-

[56] Im Internet: http://www.microsoft.com/virtualearth/.

tegrierbare JavaScript-Komponente, mit deren Hilfe Informationsblöcke in Form von Mashups auf einer virtuellen Landkarte platziert und visualisiert werden können.

Für mobile Endgeräte existiert eine spezielle Version namens Virtual Earth Mobile, welche nach Fuller (2005) neben dem Abruf von Kartenmaterial, basierend auf der Spezifikation der vier Werte Latitude[57], Longitude[58], Zoomstufe und Kartentyp, auch die Kartensuche basierend auf vom Anwender spezifizierbaren Adressen oder örtlichen Umgebungsparametern, wie z.B. Restaurants, ermöglicht.

4.4.2 DinnerNow

Bei DinnerNow[59] handelt es sich nach Pickert (2007) um einen fiktiven Marktplatz zum Bezug von Lebensmitteln aus der näheren Umgebung. Yong (2008) führt weiter aus, dass es sich um eine Technologie-Demo handelt, welche dazu dienen soll, aufzuzeigen, wie vernetzte Anwendungen auf Basis neuer Microsoft-Technologien realisiert werden können. Nach Pickert (2007) und Yong (2008) kommen dabei primär folgende Technologien zum Einsatz:

- Internet Information Services 7 (IIS 7)
- Active Server Pages .NET Ajax Extensions (ASP .NET Ajax Extensions)
- Language Integrated Query (LINQ[60])
- Windows Communication Foundation (WCF[61])
- Windows Workflow Foundation (WF)
- Windows Presentation Foundation (WPF)
- Windows PowerShell (PSH)
- .NET Compact Framework (.NET CF)

Der DinnerNow-.NET Compact Framework-Client für mobile Endgeräte unterstützt dabei unter Anderem die Benachrichtigung des zum Kunden örtlich nahen, mobilen Essensauslieferers in Form von Informationen auf dessen Windows Mobile-Empfangsgerät. Neben reiner Information in Form von Quell- und Zieladresse können Aufträge vom mobilen Anwendern explizit angenommen werden, was über WCF eine Rückmeldung an den zentralen DinnerNow-Server nach sich zieht.

4.4.3 Trazers

Ferner existiert mit Trazers[62] eine Open Source-Software, die nach Smith (2007) als Beitrag für den Microsoft Imagine Cup 2006 [63] entwickelt wurde und sich originär das Ziel gesetzt hat, die Mobilität von behinderten Menschen und dabei vor allem Rollstuhlfahrern zu verbessern, indem ein System bereitgestellt wird, welches die Navigation auf barrierefreie Art ermöglicht. Daneben werden über Trazers Informationen bereitgestellt, die auch für andere Benutzergruppen wie Fahrradfahrer und Läufer nützlich sind.

[57] Latitude = geographische Breite.
[58] Longitude = geographische Länge.
[59] Im Internet: http://www.dinnernow.net/.
[60] Vgl. Kapitel 2.2.4.1.
[61] Vgl. Kapitel 2.2.4.2.
[62] Im Internet: http://www.codeplex.com/trazers.
[63] Im Internet: http://www.microsoft.com/hk/msdn/ic2006/default.mspx.

Im Gegensatz zu herkömmlichen Navigationssystemen bietet Trazers nach ebd. zusätzlich zu reinen Straßendaten Informationen über die Güte der verschiedenen Strecken unter Berücksichtigung von Barrieren. Der Datenbestand wird hierbei auch nicht zentral gepflegt, sondern entsteht durch die Community, die ausgerüstet mit mobilen Endgeräten und einer Installation von Trazers automatisch dazu beiträgt, Informationen über besuchte Strecken zu sammeln, wobei die Häufigkeit der Wahl einer Strecke in die Berechnung mit einfließt.

Die Informationen, unter Anderem GPS-Daten zum aktuellen Standort des Trazers-Anwenders, werden in definierten Zeitintervallen an einen zentralen Server zur Auswertung übermittelt, oder bei ortsabhängig schlechter Verbindung lokal gesammelt und dann im Bulk[64]-Betrieb bei verfügbarer Netzverbindung übermittelt. Alle Teilnehmer dieses Projektes können dann bei Durchführung eigener Streckenbegehungen sich an den gesammelten Daten anderer Nutzer orientieren und Hinweise über gegebenenfalls vorliegende Barrieren abrufen. Auch besteht die Möglichkeit, weitere nützliche Metainformationen zu Strecken zu hinterlegen, wie z. B. Fotographien.

4.4.4 Griffin Navigator

Bei Griffin Navigator[65] handelt es sich um ein klassisches GPS-basiertes Navigationssystem, welches auf dem .NET Framework in der Version 2.0 basiert und Kartenmaterial des Microsoft Virtual Earth Tile System einsetzt.

Daten werden über das Microsoft Virtual Earth Tile System online bezogen, können aber auch in einer lokalen SQL Server 2005 Compact Edition-Datenbank gespeichert werden.

4.4.5 SniffThat

SniffThat[66] ist eine .NET Compact Framework-basierte Anwendung zum Scannen nach drahtlosen Zugriffspunkten (Access Points) und zur Vermengung dieser Informationen mit aktuellen GPS-Daten des mobilen Empfängers sowie zur Visualisierung dieser Daten.

Nach Lamers-Software (o.J.) unterstützt die Architektur von SniffThat beliebige Ex- und Importmodule um die von Access Points stammenden Daten und GPS-Daten sowie optional Zusatzinformationen aus Geoinformationsdatenbanken zu verarbeiten. Nach ebd. sind folgende Module standardmäßig in SniffThat enthalten:

- XML Export (Exportmodul)
- Ov2 Export (TomTom POI Format) (Exportmodul)
- Kml Export (Google Earth Format) (Exportmodul)
- Access Point Liste (Visuelles Modul)
- Radar (Visuelles Modul)
- Gps Informationen im Umkreis (Visuelles Modul)
- Gps Status (Visuelles Modul)

[64] Übertragung einer größeren, gesammelten Datenmenge im Rahmen eines einzelnen Vorgangs.
[65] Im Internet: http://www.codeplex.com/griffinnav.
[66] Im Internet: http://www.codeplex.com/sniffthat.

4.5 Konklusion

In diesem Kapitel wurde die Unterstützung von ortsabhängigen Diensten im .NET Compact Framework und der damit häufig in Verbindung genutzten Technologien untersucht.

Wie sich gezeigt hat, liegt eine breite Unterstützung sowohl auf Betriebssystemebene, Frameworkebene als auch auf Datenpersistierungsebene vor, wenngleich noch keine verwaltete Unterstützung zum Zugriff auf Positionsinformationen durch das .NET Compact Framework selbst besteht. Daneben existiert bereits eine Vielzahl von Anwendungen, welche gezielt Funktionalität ortsabhängiger Dienste nutzen, wie DinnerNow und Trazers.

5 Fazit

Zusammengefasst kann festgehalten werden, dass das .NET Compact Framework eine umfangreiche und hochflexible Architektur für mobile Geräte auf Basis von Windows Mobile-Betriebssystemen bietet.

Neben den frameworkspezifischen Funktionalitäten wie der Anbindung nahezu aller relevanten DBMS, der komfortablen Selektion und Verarbeitung von Daten über LINQ und der flexiblen Unterstützung beliebiger verteilter Dienste über WCF, besteht auch eine gute Integration in die Entwicklungsumgebung Visual Studio 2008. Diese IDE bietet dabei eine Vielzahl von Werkzeugen zur komfortablen Entwicklung und umfangreichen Prüfung der zu erstellenden mobilen Applikationen.

Ferner besteht eine gute Unterstützung ortsabhängiger Dienste. Dies schließt primär

- auf Betriebssystemebene den Bereich Simulierung von ortsabhängigen Umgebungsparametern und des GPS-Empfängers,
- auf Frameworkebene die GPS-Kommunikation, die Anbindung bestehender Dienste und die Nutzung von Geoinformationssystemen samt zugehöriger Berechnungen,
- auf Datenpersistierungsebene die Unterstützung räumlicher Datentypen und Synchronisierungsdienste
- und auf Anwendungsebene die Möglichkeiten der Nutzung von Online-Kartendiensten und Navigationssystemen mit ein.

Einzig die Beschränkung auf Windows Mobile-basierte mobile Geräte, die noch fehlende Möglichkeit eines verwalteten Zugriffs auf alle Positionsbestimmungshardwarekomponenten sowie die derzeitig aus Performancegründen noch vorliegenden Einschränkungen von LINQ und WCF auf mobilen Endgeräten sind als negative Aspekte zu bewerten. Aufgrund der fortschreitenden Leistungsfähigkeit mobiler Hardware sollten die LINQ- und WCF-bezogenen Beschränkungen jedoch in mittelfristiger Zukunft aufgehoben werden können.

Literaturverzeichnis

[Arnott 2007] Arnott, Andrew: The WCF subset supported by NetCF. 2007. – URL http://blogs.msdn.com/andrewarnottms/archive/2007/08/21/the-wcf-subset-supported-by-netcf.aspx. – Letzter Zugriff am 27.04.2008

[Aschenbrenner 2008] Aschenbrenner, Klaus: Stadt, Land, Fluss mit SQL: Spatial Data Types beim SQL Server 2008. In: *dotnetpro* 05 (2008), S. 108–115

[Bruchmann 2007] Bruchmann, Kai: Beast Master.Net. 2007. – URL http://www.pocketkai.net/asp/en/index-en.html?seitex=http://www.pocketkai.net/asp/en/details.aspx?338. – Letzter Zugriff am 03.05.2008

[Celarier und Creek 2005] Celarier, Stuart ; Creek, Fern: .NET Compact Framework FAQ - .NET Compact Framework (General). 2005. – URL http://msdn2.microsoft.com/de-de/netframework/aa497275(en-us).aspx. – Letzter Zugriff am 22.04.2008

[Chopra 2007]Chopra, Amit: New Device Development Features in Visual Studio 2008. 2007. – URL http://iphone.sys-con.com/read/478947_p.htm. – Letzter Zugriff am 28.04.2008

[ComponentSource o.J.] ComponentSource (Hrsg.): *Database Connectivity / .NET Compact Framework*. o.J.. – URL http://www.componentsource.com/features/database-connectivity/net-compact-framework/index.html. – Letzter Zugriff am 27.04.2008

[DotNET Compact Framework Team 2007] DotNET Compact Framework Team (Hrsg.): *DotNet CF 2.0 SP2 Released*. 2007. – URL http://blogs.msdn.com/netcfteam/archive/2007/03/13/net-compact-framework-2-0-sp2-released.aspx. – Letzter Zugriff am 24.04.2008

[dotnet.de 2007] dotnet.de (Hrsg.): *.NET 3.5 und Visual Studio 2008 sind da*. 2007. – URL http://entwickler.com/itr/news/psecom,id,38670,nodeid,82.html. – Letzter Zugriff am 24.04.2008

[Esposito 2005] Esposito, Dino: *Programming Microsoft ASP.NET 2.0 Core Reference*. Redmond, WA, USA : Microsoft Press, 2005

[Feinman 2006] Feinman, Alex: Debuggen und Emulieren von Geräten in Visual Studio 2005. 2006. – URL http://msdn2.microsoft.com/de-de/library/aa454884.aspx. – Letzter Zugriff am 29.04.2008

[Fuller 2005] Fuller, Jason: Virtual Earth Mobile. 2005. – URL http://blogs.msdn.com/windowsmobile/archive/2005/10/23/VEMobile.aspx. – Letzter Zugriff am 01.05.2008

[Fulton 2008] Fulton, Scott M.: CTP for SQL Server 2008 now available. 2008. – URL http://www.betanews.com/article/CTP_for_SQL_Server_2008_now_available/1203538089. – Letzter Zugriff am 01.05.2008

[Gamma et al. 1995] Gamma, Erich ; Helm, Richard ; Johnson, Ralph E.: *Design Patterns. Elements of Reusable Object-Oriented Software.* 1st ed., Reprint. Addison-Wesley Longman, Amsterdam, 3 1995

[GSM Association 2008] GSM Association (Hrsg.): *About the GSM Association*. 2008. – URL http://www.gsmworld.com/about/index.shtml. – Letzter Zugriff am 18.05.2008

[György 2008] György, Balássy: NXT.NET for LEGO Mindstorms. 2008. – URL http://www.codeplex.com/NxtNet. – Letzter Zugriff am 03.05.2008

[Hart 2007] Hart, Simon: What's new in .NET 3.5 CF. 2007. – URL http://simonrhart.blogspot.com/2007/12/whats-new-for-device-developers-in.html. – Letzter Zugriff am 24.04.2008

[Hossain 2007] Hossain, Mehfuz: LINQ provider basics. 2007. – URL http://weblogs.asp.net/mehfuzh/archive/2007/10/04/writing-custom-linq-provider.aspx. – Letzter Zugriff am 30.04.2008

[Hubschneider und Kölmel 2002] Hubschneider, Martin ; Kölmel, Bernhard: Mobile Business - Location Based Services als Killerapplikation. 2002. – URL http://www.e-lba.com/YellowMap_Location%20Based%20Services%20als%20Killerapplikation.pdf. – Letzter Zugriff am 02.05.2008

[IC#Code o.J.] IC#Code (Hrsg.): *#develop Features*. o.J.. – URL http://www.icsharpcode.net/OpenSource/SD/Features.aspx. – Letzter Zugriff am 28.04.2008

[IDG Business Media GmbH 2005] IDG Business Media GmbH (Hrsg.): *Mobile Dienstleistungen werden sich bis 2010 verdoppeln*. 2005. – URL http://www.channelpartner.de/_misc/article/print/index.cfm?pid=143&pk=225255&op=prn. – Letzter Zugriff am 02.05.2008

[Klein 2008] Klein, Scott: *Professional LINQ (Programmer to Programmer)*. 1. Wiley & Sons, 1 2008

[Küpper 2005] Küpper, Axel: *Location-Based Services: Fundamentals and Operation*. Wiley, 10 2005

[Kuhbach 2003] Kuhbach, Sebastian: .NET Compact Framework 1.0 Redistributable - Download inside. 2003. – URL http://winfuture.de/news-druckansicht.php?id=8140. – Letzter Zugriff am 23.04.2008

[Kuhrmann und Beneken 2006] Kuhrmann, Marco ; Beneken, Gerd: *Windows® Communication Foundation: Konzepte - Programmierung - Migration*. Spektrum Akademischer Verlag, 11 2006

[Lamers-Software o.J.] Lamers-Software (Hrsg.): *SniffThat: Beschreibung*. o.J.. – URL http://www.lamesoft.de/Lamers-Software/Projekte/SniffThat/tabid/77/Default.aspx. – Letzter Zugriff am 03.05.2008

[Louis und Müller 2008] Louis, Dirk ; Müller, Peter: *Java 6 Kompendium*. 1. Markt und Technik, 2008

[Löwy 2007] Löwy, Juval: *Programming WCF Services (Programming)*. 1. O'Reilly Media, 2 2007

[Manzoor 2002] Manzoor, Kashif: The Common Language Runtime (CLR) and Java Runtime Environment (JRE). 2002. – URL http://www.codeproject.com/KB/dotnet/clr.aspx?display=Print. – Letzter Zugriff am 21.04.2008

[Microsoft Corporation 2003] Microsoft Corporation (Hrsg.): *Hospital Improves Patient Assessment Efficiency with .NET Compact Framework*. 2003. – URL http://download.microsoft.com/documents/australia/business/casestudies/rmh.pdf. – Letzter Zugriff am 03.05.2008

[Microsoft Corporation 2007] Microsoft Corporation (Hrsg.): *What Is Windows Communication Foundation?* 2007. – URL http://msdn2.microsoft.com/en-us/library/ms731082(printer).aspx. – Letzter Zugriff am 27.04.2008

[Microsoft Corporation o.J.a]Microsoft Corporation (Hrsg.): *.NET Compact Framework for Xbox 360*. o.J. – URL http://msdn2.microsoft.com/en-us/library/bb203912.aspx. – Letzter Zugriff am 22.04.2008

[Microsoft Corporation o.J.b] Microsoft Corporation (Hrsg.): *Devices and platforms supported by .NET CF*. o.J. – URL http://msdn2.microsoft.com/en-us/library/ms172550(printer).aspx. – Letzter Zugriff am 26.04.2008

[Microsoft Corporation o.J.c]Microsoft Corporation (Hrsg.): *NET Compact Framework Data Providers*. o.J. – URL http://msdn2.microsoft.com/en-us/library/aa275630(SQL.80).aspx. – Letzter Zugriff am 27.04.2008

[Microsoft Corporation o.J.d] Microsoft Corporation (Hrsg.): *Verwenden von SQL Server Compact 3.5-Datenbanken (Geräte)*. o.J. – URL http://msdn2.microsoft.com/de-de/library/aa983313(printer).aspx. – Letzter Zugriff am 27.04.2008

[Microsoft Corporation o.J.e]Microsoft Corporation (Hrsg.): *Introduction to Occasionally Connected Applications using MSF*. o.J. – URL http://msdn.microsoft.com/en-us/sync/bb887608.aspx. – Letzter Zugriff am 02.05.2008

[Mosa 2006] Mosa, Muhammad: Abstract Factory Design Pattern in ADO.NET 2.0. 2006. – URL http://www.c-sharpcorner.com/UploadFile/mosessaur/abstractfactoryadonet20215200605364 3AM/abstractfactoryadonet2.aspx?ArticleID= 4468e7cc-57c1-4738-8e83-a8db48fd1d9f. – Letzter Zugriff am 27.04.2008

[Nielsen 2007a] Nielsen, Morten: SharpGPS: Project description. 2007. – URL http://www.codeplex.com/SharpGPS. – Letzter Zugriff am 01.05.2008

[Nielsen 2007b] Nielsen, Morten: SharpGPS - Release: 0.5 alpha. 2007. – URL http://www.codeplex.com/SharpGPS/Release/ProjectReleases.aspx?ReleaseId=5048. – Letzter Zugriff am 03.05.2008

[Pialorsi und Russo 2007] Pialorsi, Paolo ; Russo, Marco: *Introducing Microsoft LINQ*. Microsoft Press, 2007

[Pickert 2007] Pickert, Hardy: Adventures in .NET. 2007. – URL http://hardypickert.de/blog/2007/02/01/DinnerNowNETSampleApplication.aspx. – Letzter Zugriff am 02.05.2008

[Pohmann 2004] Pohmann, Peter: Das ADO.NET-DataSet. 2004. – URL http://msdn2.microsoft.com/de-de/library/bb979079(printer).aspx. – Letzter Zugriff am 27.04.2008

[Pratschner 2004] Pratschner, Steven: .NET Compact Framework Garbage Collector. 2004. – URL http://blogs.msdn.com/stevenpr/archive/2004/07/26/197254.aspx. – Letzter Zugriff am 26.04.2008

[Roof 2003] Roof, Larry: Pocket Access and the .NET Compact Framework. 2003. – URL http://msdn2.microsoft.com/en-us/library/ms837914(printer).aspx. – Letzter Zugriff am 27.04.2008

[Rubin und Yates 2003] Rubin, Erik ; Yates, Ronnie: *Microsoft .NET Compact Framework Kick Start*. Sams, 2003

[Schmidt und Laerhoven 2001] Schmidt, Albrecht ; Laerhoven, Kristof V.: How to build smart appliances. In: *IEEE Personal Communications* (2001), S. 66–71

[Smith 2007] Smith, Jack: Mobile GPS Navigation for the Web: Project Description. 2007. – URL http://www.codeplex.com/trazers. – Letzter Zugriff am 02.05.2008

[Troelsen 2007] Troelsen, Andrew: *Pro C# 2008 and the .NET 3.5 Platform, Fourth Edition (Pro Series)*. Berkely, CA, USA : Apress, 2007

[Watt 2007] Watt, Andrew: *Professional Windows PowerShell*. Birmingham, UK, UK : Wrox Press Ltd., 2007

[Weinberg 2008] Weinberg, Nathan: Exclusive Windows Mobile 7 To Focus On Touch and Motion Gestures. 2008. – URL http://microsoft.blognewschannel.com/archives/2008/01/06/exclusive-windows-mobile-7-to-focus-on-touch-and-motion-gestures/. – Letzter Zugriff am 13.05.2008

[Wells 2005] Wells, Jonathan: .NET Compact Framework 1.0 SP3 Developer Redistributable. 2005. – URL http://blogs.msdn.com/onoj/archive/2005/04/29/413409.aspx. – Letzter Zugriff am 24.04.2008

[Widmann 2008] Widmann, Britta: Microsoft präsentiert Windows Mobile 6.1. 2008. – URL http://www.zdnet.de/news/tkomm/0,39023151,39189077,00.htm. – Letzter Zugriff am 13.05.2008

[Wigley et al. 2003] Wigley, Andy ; Wheelwright, Stephen ; Burbidge, Robert ; MacLoed, Rory ; Sutton, Mark: *Microsoft .NET Compact Framework (Core Reference)*. 1. Microsoft Press, 1 2003

[Wildermuth 2005] Wildermuth, Shawn: Choosing a Database for Compact Framework Applications. 2005. – URL http://www.devsource.com/c/a/Techniques/Choosing-a-Database-for-Compact-Framework-Applications/. – Letzter Zugriff am 27.04.2008

[Wilson 2005a] Wilson, Jim: What's New in the .NET Compact Framework 2.0. 2005. – URL http://msdn2.microsoft.com/en-us/library/aa446574(printer).aspx. – Letzter Zugriff am 24.04.2008

[Wilson 2005b] Wilson, Jim: .NET Compact Framework 2.0 Supported Platforms. 2005. – URL http://www.pluralsight.com/blogs/jimw/archive/2005/11/14/16776.aspx. – Letzter Zugriff am 26.04.2008

[Wilson 2007] Wilson, Jim: Windows Mobile 6 - Neuigkeiten für Entwickler. 2007. – URL http://www.microsoft.com/germany/msdn/library/mobility/windowsmobile/WindowsMobile6NeuigkeitenFuerEntwickler.mspx?mfr=true. – Letzter Zugriff am 01.05.2008

[WindowsForDevices.com 2005] WindowsForDevices.com (Hrsg.): *Microsoft releases .NET CF 2.0 redistributable*. 2005. – URL http://www.windowsfordevices.com/news/NS6191688737.html. – Letzter Zugriff am 24.04.2008

[Yong 2008] Yong, Patrick: Dinner Now project for VS2008 RTM out! 2008. – URL http://patrickyong.net/2008/03/04/dinner-now-project-for-vs2008-rtm-out/. – Letzter Zugriff am 03.05.2008

Abbildungsverzeichnis

Listingverzeichnis