Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Diplomarbeit

Untersuchung von Mono als Plattform für Webservices auf mobilen Endgeräten

vorgelegt von
Lars Mählmann
am 28. April 2005

Studiengang Softwaretechnik

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Dipl. Inf. Birgit Wendholt

Fachbereich Elektrotechnik und Informatik
Department of Electrical Engineering and Computer Science

Untersuchung vonMono

Stichworte Ubiquitous Computing, Bluetooth, Web-Services, Handhelds, Linux, Mono, Middleware , Verteilte Systeme

Zusammenfassung

Durch die steigenden Anforderungen an die Software in den verschiedenen Anwendungsgebieten wandeln sich die Werkzeuge zur Softwareentwicklung. Um diesen gerecht zu werden, wurde mit Java eine plattformunabhängige Programmiersprache entwickelt, die auf verschiedenen Systemen anwendbar ist. Microsoft hat mit .NET versucht unterschiedliche Programmiersprachen auf einer gemeinsamen Plattform zu verbinden. Die zur Zeit geforderten Ziele nach verteilten Anwendungen und Komponenten soll mit Techniken wie Corba oder Web-Services realisiert werden. Die sprachunabhängige Plattform .NET ist Web-Service orientiert und versucht so diesen Anforderungen gerecht zu werden. Mit Mono wurde zusätzlich zu der Sprachunabhängigkeit von .NET eine Plattformunabhängigkeit versucht zu realisieren. In dieser Arbeit soll evaluiert werden, wie gut Mono als Plattform für Web-Services auf mobilen Endgeräten einsetzbar ist. Mit Hilfe einer realistischen Umgebung soll Mono untersucht werden. Abschliessend sollen die Ergebnisse dargestellt und disskutiert werden.

Examination of Mono

Keywords Ubiquitous Computing, Bluetooth , Web-Services, Handhelds, Linux , Mono, Middleware, Distributed Systemes

Abstract

The tools for software development have changed because of increasing requirements of software in the different areas of application. Java has been developed with a focus on a platform independent programming language. Microsoft has tried to combine different programming languages on one plattform, called .NET. The required target of distributed applications and components should be realised with techniques like cobra and web-sevices. The language independence plattform .NET is oriented on web-services and tries to meet the requirements. Additionaly to the language independence outside of Windows Mono tried to realize a platform independence. In this thesis Mono will be evaluated as a plattform for web-services on mobile end devices. Mono will be examined in a realistic environment. Finally the results are displayed and discussed.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

In der heutigen Zeit kann beobachtet werden wie sich die aktuellen Entwicklungsplattformen von einer “einfachen” Programmiersprache zu einer modernen Plattform wandeln. Auf diesen sollen sämtliche Anforderungen und Probleme gelöst werden, unabhängig von der verwendeten Betriebssystemplattform (z. B. Linux, Windows oder MacOS) oder dessen Hardware (z. B. Apple, PC, mobile Endgeräte oder Handys). Bei diesem Wandel stehen allgemeine Anforderung, wie Ubiquitous Computing oder Verteilte Systeme, im Vordergrund. Aber was verbirgt sich dahinter: Der Begriff *Ubiquitous Computing* meint die Allgegenwärtigkeit der Informationsverarbeitung. Der Begriff wurde von Mark Weiser in seinem Aufsatz “The Computer for the 21st Century” geprägt. Nach seiner Vision wird der (Personal-) Computer als Gerät verschwinden, bzw. durch ïntelligente Gegenstände", ersetzt werden, die den Menschen bei seinen Tätigkeiten unterstützt [Weiser 1991].
Der zweite Punkt sind Verteilte Systeme, bzw. Anwendungen. Es soll Netzwerken, wie Mobiltelefon-, Unternehmens-, Universitäts- oder privaten Netzwerken, die Möglichkeit gegeben werden, sich zu verbinden und somit die gemeinsame Nutzung von Ressourcen, die beliebige räumliche Distanzen aufweisen, zu nutzen. Internationalisierung ist eine weitere Anforderung, die durch den weltweit zunehmenden Handel mit Hilfe von Computern mehr und mehr an Bedeutung gewinnt. Dabei möchte man zusätzlich die bisherige monolithische Architektur [Wikipedia 2005] ersetzen und immer nur die für eine Anwendung erforderlichen Anteile einbinden, bzw. zur Verfügung stellen. Ein Beispiel dafür wären SOA (Service-Orientierte Architekturen). Durch die mit den Jahren immer leistungsfähiger gewordenen mobilen Geräte, wie PDAS und Handys, werden immer mehr Anwendungen von Personalcomputern auf mobile Geräte exportiert. Die Geräte können Aufgaben übernehmen, die sie früher nicht bewältigen konnten. Es ist z. B. möglich, dass Sachverständige bei der Abnahme von Fahrzeugen ein mobiles Gerät zur Dokumentation benutzen und somit die entsprechenden Daten digital vorliegen haben, um diese später direkt weiter zu verwenden.
Ein letzter Punkt sind die sich ändernden Geschäftsprozesse in Unternehmen.
Es muss alles schnell und flexibel änderbar sein, wie z. B. der Handel mit fremden Unternehmen. Für alle diese Forderungen gilt eine allgemeingültige Verfügbarkeit auf vielen unterschiedlichen Systemen und Programmiersprachen anzubieten.

Um diesen Anforderungen gerecht zu werden, verwenden viele Entwickler überwiegend zwei Plattformen. Java ist eine von Sun entwickelte betriebssystemunabhängige Programmiersprache mit der es möglich ist Anwendungen zu schreiben, die auf den unterschiedlichsten

Betriebssystemen laufen. Java erreicht dieses durch die Definition einer eigenen Plattform, die auf dem Zielsystem durch eine Java Virtual Machine Java (VM) implementiert wird. Diese Plattform bietet, verschiedene Bibliotheken an mit denen eigene Anwendungen realisiert werden können. Der Vorteil dabei ist die eigene Laufzeitumgebung, die es erlaubt, unabhängig von dem verwendeten Betriebsystem zu entwickeln. Die einzige Voraussetzung ist eine Java VM auf den jeweiligen Systemen. Deren Verwendung kann allerdings bei geringen Ressourcen (z. B. mobiles Endgerät) zu Problemen führen. Im Gegensatz zu .NET ist die Programmiersprache Java bindend und kann nicht erweitert oder mit anderen Sprachen kombiniert werden.

Bei der von Microsoft entwickelten Plattform .NET handelt es sich um eine relativ sprachunabhängige Plattform[1], die es erlaubt verschiedene Programmiersprachen in einer Anwendung zusammen zu nutzen. Der Vorteil für den Entwickler ist, dass er unabhängig von den bisher benutzten Umgebungen verschiedene Entwicklungen kombinieren kann. Für den Anwender ist es somit möglich verschiedene einzelne Anwendungen zusammen zu nutzen, wodurch sich die Arbeit vereinfachen oder effektiver gestalten lassen kann. Der hier entstandene Nachteil ist, dass diese Plattform nur in Zusammenhang mit einem Microsoft Betriebssystem lauffähig ist. Somit stehen die Vorteile dieser Plattform den Anbietern von Microsoft fremden Systemen (z. B. Linux, MacOS, Unix) nicht zur Verfügung.

Eine mögliche Lösung dieser Problematik könnte Mono sein. Das von Miguel de Icaza ins Leben gerufene Projekt versucht die wesentlichen Bestandteile von .NET auf anderen Betriebssystemen lauffähig anzubieten, um die Sprachunabhängigkeit von .NET auf weiteren Plattformen zu ermöglichen. Microsoft hat .NET und die Sprache C# als offenen Standard bei der Standardisierungsorganisation ECMA[11] eingereicht. Damit war es möglich dieses Projekt zu realisieren. Mit Mono ist eine sprachunabhängige Plattform erhältlich, die auf verschiedenen Betriebssystemen lauffähig ist. Mono verbindet scheinbar die Vorzüge von Java und .NET.

In dieser Arbeit findet eine Evaluation von Mono an einem repräsentativen Beispiel statt. Es wird untersucht, ob Mono den Anforderung an eine moderne Plattform nachkommt. Die Verfügbarkeit des Frameworks wird hinsichtlich der Einsetzbarkeit in einem Verteilten System, auf verschiedenen Endgeräten und Betriebssystemen untersucht.

[1] die Sprachen, die im Moment unterstützt werden sind sich von ihrer Syntax ähnlich (C#, C++, C und VS)

Gliederung der Arbeit

Im ersten Abschnitt der Arbeit soll die Aufgabenstellung beschrieben werden in der Mono untersucht wird. Hierbei soll ein realistisches Szenario mit daraus resultierenden Anforderungen entwickelt werden, um Mono zu testen.
Die entstehenden Probleme und Anforderungen werden unter Beachtung des Benutzers und der Informatik im Kapitel 3 beschrieben werden. Auf Grund des Szenarios soll die Middleware hervorgehoben werden, da diese die Schnittstelle zwischen den einzelnen Komponenten bieten muss. Nachdem das Szenario skizziert, die Anforderung beschrieben und die speziellen Anforderungen an die Middleware dargestellt wurden, soll überlegt werden, wie mögliche Geschäftsmodelle (Kapitel 3.4) aussehen könnten, um diese in den Entwurf mit einzubeziehen. Durch die Darstellung möglicher Geschäftsprozesse erhält man ein besseres Verständniss für das Szenario und die Anforderungen. Dies trägt zu einer besseren Lösungsfindung bei.
Im Entwurf (s. Kapitel 4) wird eine grundsätzliche Lösungsstrategie vorgestellt, die auf die vorher geforderten Bedingungen und Geschäftsabläufe Bezug nimmt. In einem ersten Schritt wird eine Skizze der Komponenten und deren Zusammenhang dargelegt. Diese Darstellung vermittelt eine erste Idee des Systementwurfs. Daraus wird die Architektur entwickelt und die einzelnen Systemkomponenten vorgestellt. Diese sollen mit Hilfe eines Geschäftsablaufes verbessert werden. Mit einem Sequenzdiagramm wird der Ablauf überprüft und erläutert. Die einzelnen Systemkomponenten werden mit dessen Hilfe verfeinert. Unter Zuhilfenahme von bekannten Designmustern soll der bestehende Entwurf überprüft und verbessert werden. Sind die Anforderungen aus dem Szenario in den Entwurf eingeflossen und die Systemarchitektur sowie die Komponenten bestimmt, sollen diese in den darauffolgenden Kapiteln in einen Versuchsaufbau umgesetzt werden. Der Entwurf dient einer prototypischen Umsetzung, mit der Mono untersucht werden soll.
Bei der Realisierung wird zunächst der Versuchsaufbau erläutert. Für das bessere Verständniss werden die einzelnen Technologien im Kapitel 5 vorgestellt und es erfolgt eine Einführung. Zusätzlich wird die Laborumgebung genauer spezifiziert. Hierbei soll ein besseres Verständniss für die Umgebung vermittelt werden. Am Ende des Kapitels 6.2 werden die Erfahrungen aus der Implementierung des Prototypen zusammengefasst und ein Fazit gezogen, inwieweit der entwickelte Entwurf umsetzbar war und worin die Stärken und Schwächen von Mono bestanden.
Als letztes werden im Kapitel 7 die im Rahmen der Arbeit gesammelten Erkenntnisse dieser Evaluierung zusammengefasst und diskutiert. Das Verhältnis zwischen dem Projekt Mono und Microsoft soll zusätzlich angesprochen werden. Als Abschluss soll ein kleiner Blick in die Zukunft von Mono gewagt werden.

2. Szenario

Im folgendem soll Mono in Anlehnung an ein bereits von IBM [Snell 2002] beschriebenes Szenario untersucht werden. Dazu dient ein fiktiver Ferienclub. Untersucht werden soll, ob Mono eine geeignete Plattform ist, um die Anforderung verschiedener Systeme in einer verteilten Anwendung zu erfüllen. Ein PDA dient als “rich client”, um Angebote von verschiedenen Anbietern in einem Ferienclub zu buchen. Es wird dabei eine Anwendung benutzt, die über Web-Services auf die verschiedenen Angebote zugreifen kann und diese bucht.

Das Testszenario für Mono wird skizziert. Es folgt die Beschreibung des fiktiven Ferienclubs und den dazugehörigen Anbietern. Ebenfalls folgt eine Vorstellung des Clients, mit welchem die Gäste in diesem Ferienclub ausgestattet werden. Weiterhin soll erklärt werden, wie die Freizeitangebote für den Gast nutzbar gemacht und wie die verschiedenen Techniken hier eingesetzt werden können.

2.1. Einzelne Anwendungsfälle

Anwendungsfall Gast - Ferienclub

Bei der Ankunft des Gastes im Hotel und Abarbeitung der Formalitäten der Anmeldeprozedur wird dem Gast ein mobiles Gerät (Personal Digital Assistent (PDA)) übergeben, der mit einer Funkverbindung ausgestattet ist, durch die das Gerät auf dem Feriengelände mit dem Hotel verknüpft ist. Auf dem Gerät befindet sich ein Anwendung mit welcher der Gast nun die verschiedenen Freizeitangebote abrufen und auf dem Gerät verwalten kann. Wenn der Gast sich im Funknetz des Hotels befindet, sollen die Angebote automatisch durch die Anwendung mit dem Hotelserver synchronisiert werden. Es kann aber auch möglich sein, dass der Gast die Synchronisation selbst anstößt. Sobald die Angebote auf dem PDA vorliegen, wird die Funkverbindung nicht mehr zwingend benötigt. Die Angebote können “offline” gelesen werden. Dem Gast muss die Aktualität des Angebote mitgeteilt werden. Die Anwendung soll den Gast daran erinnern die Angebote zu aktualisieren, wenn die letzte Synchronisation zu weit zurück liegt. Bei Interesse kann dann offline die Reservierungsanfrage vorbereitet werden und sobald der PDA wieder eine Verbindung zum Hotelnetz erhält, wird die Anfrage verschickt. Der Reservierungswunsch wird vom Hotel angenommen und weiterbearbeitet.

Bevor die Buchung rechtsgültig wird, erhält der Gast die Aufforderung die gesamte Buchung zu bestätigen. Möchte der Gast die Reservierung stornieren oder ändern, soll das mit Hilfe der Reservierungsbestätigung erfolgen, wenn alle rechtlichen Anforderung (z. B. Fristen, Gebühren) erfüllt sind (siehe Abbildung 2.1). Es ist dabei wichtig, dass der Gast die Stornierung nicht nur über den PDA durchführen kann, sondern zusätzlich auch beim Hotelpersonal direkt. Das Hotel hat die Möglichkeit eine Reservierung direkt auf dem Ferienclubserver zu stornieren oder zu ändern, ohne dass der Gast einen derartigen Wunsch über den PDA abschicken muss.

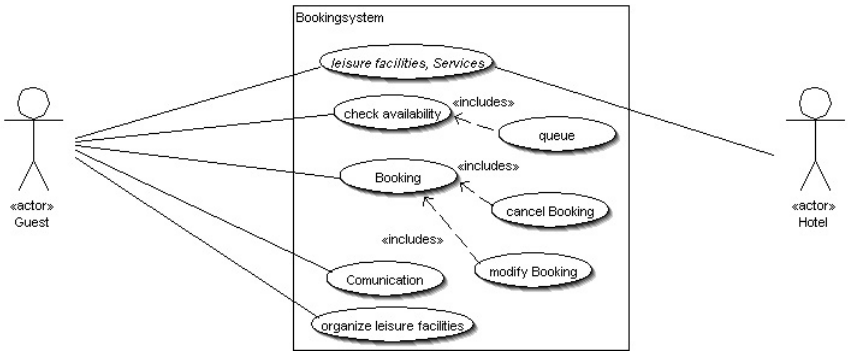

Abbildung 2.1.: Angebote an den Gast um Reservierungen zu buchen

Anwendungsfall Ferienclub - Anbieter

Wenn der Ferienclub und ein Anbieter, z. B. Autovermietung, sich auf eine Kooperation geeinigt haben, muss der Anbieter dem Hotel eine Schnittstelle zur Verfügung stellen mit der dieser die Angebote abholen kann. Diese ist unabhängig vom jeweils verwendeten System und ist vom Hotelanbieter in das eigene System zu integrieren. Der Anbieter bereitet die Leistungen für das Hotel auf und bietet sie dem Hotel über die definierte Schnittstelle an. Das Hotel holt die Angebote täglich ab und aktualisiert dann die vorhandenen Angebote mit den neueren. Buchungsanfrage vom Gast, prüft das Hotel auf Vollständigkeit und Korrektheit. Es fragt dann den Anbieter, ob die Anfrage gebucht werden kann. Der Anbieter prüft das Angebot und macht, wenn möglich, eine Zusage. Der Ferienclub organisiert die weiteren Abläufe mit dem Gast und stellt die Rechnung an den Gast. Die Rechnung wird dem Gast auf seine Gesamtrechnung hinzugefügt und bei Abreise bezahlt. Die Anbieter stellen eine Rechnung über alle Buchungen in einem festgelegten Zeitraum. Das Hotel bezahlt diese Rechnung für alle Gäste (siehe Abbildung 2.2).

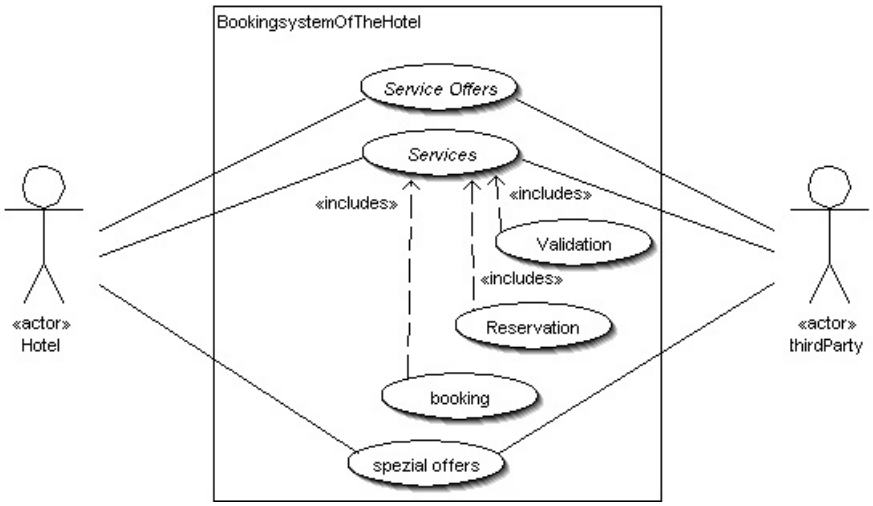

Abbildung 2.2.: Ablauf zwischen Hotel und Drittanbietern

2.2. Akteure/Rollen

Das Szenario des Ferienclubs hat verschiedene Akteure die hier interagieren. Die verschiedenen Akteure werden kurz vorgestellt, was sie darstellen und wie sie mit den jeweiligen anderen kommunizieren.

Ferienclub

Dem Ferienclub geht es um die Erweiterung der Freizeitangebote. Die bisherigen Möglichkeiten zur Freizeitgestaltung, die der Club dem Gast zur Verfügung stellen konnte, wurden immer direkt vom Hotel veranstaltet. Um nun dem Gast die Vielzahl von Angeboten aus der Umgebung zugänglich zu machen, sollen die Anbieter von Kultur (Stadtführungen), Sportangeboten (z. B. Tennis, Tauchen), Restaurants und Autovermietungen in die Angebotsliste mit aufgenommen werden.

Das Hotel handelt mit den Anbietern günstigere Konditionen für seine Gäste aus, wenn über sie bei dem jeweiligen Anbieter gebucht wird. Für den Anbieter liegt der Vorteil in der Werbung und der größeren Auslastung, die entsteht, wenn ein Hotel seinen Gästen direkt dieses Angebot unterbreitet. Der Vorteil für den Gast besteht darin, dass er mit dem Hotel einen direkten Ansprechpartner hat, dem er seine Wünsche mitteilt und der sich um alles kümmert (Abbildung 2.3).

Der Ferienclub muss sämtliche Funktionalitäten bereitstellen, um dem Gast die Reservierung von Angeboten zu ermöglichen. Dies beinhaltet die Pflege der Angebote, die Reservierung

und die Abrechnung mit Gast und Anbieter.
Zusätzlich wird eine gemeinsame Kommunikationsplattform benötigt auf der die verschiedenen Bereiche zusammenarbeiten.

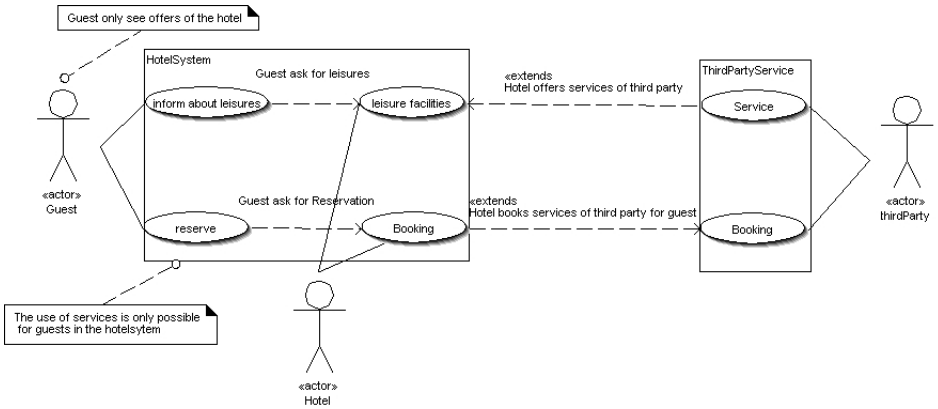

Abbildung 2.3.: Kommunikation zwischen Hotel, Drittanbietern und Gast

Administration

Der Ferienclub muss es ermöglichen dem Gast direkt zu helfen. Angenommen der Gast betätigt eine Fehlbuchung, kommt mit der Bedienung bei einer Reservierung nicht zu recht oder stellt eine Frage zum Stand seiner Reservierung oder den entstandenen Kosten, dann muss das Personal die Möglichkeit, besitzen diese Fragen zu beantworten oder aktiv die vorliegende Reservierung bearbeiten zu können. Das Personal, speziell die Verwaltung und Rezeption, hilft dem Gast im Umgang mit dem Gerät und bei den einzelnen Vorgängen der Reservierung.
Zu den aufgeführten Aufgaben gehört weiterhin die Einweisung in die Funktionsweise des Gerätes und die Abnahme des Gerätes bei der Abreise des Gastes. Bei Ausfall des Gerätes hat das Personal die Aufgabe dem Gast ein Ersatzgerät zur Verfügung zu stellen oder den Fehler direkt zu beseitigen.

Damit ein Gast alle zur Verfügung stehenden Angebote erhält, sollen diese automatisch zwischen dem Hotel und den einzelnen Anbietern abgeglichen werden. Bei diesem Abgleich ist zu unterscheiden, welche Technik benutzt werden soll. Es ist denkbar, die aktuellen Angebote nur dann abzugleichen, wenn der Gast speziell diese anfragt (On-demand) oder in festgelegten Zeiträumen die Angebote zu aktualisieren (Polling). Der Gast erhält dann alle Angebote vom Hotel über einen Freizeit- oder Terminplaner [Bresch 2004]. Bei der Reservierung bestellt der Gast beim Hotel, das den weiteren Ablauf organisiert.
Die Fremdanbieter müssen eine Möglichkeit zur Verfügung stellen, mit der die eigenen Angebote für den Ferienclub verfügbar gemacht bzw. gebucht werden können. Der Ferienclub

benutzt für die Kommunikation der mobilen Geräte spontane Netzwerkverbindungen. Der Club benötigt für die Annahme und Bearbeitung eine eigene Software und Server mit genügend Kapazität für den Service und ausserdem eine Anwendung für den Austausch mit den Anbietern.

Die Anbieter

Bei den Geschäftspartnern des Ferienclubs handelt es sich um verschiedene Unternehmen, die sich in der Touristikbranche angesiedeln haben. Die Anbieter kommen aus einzelnen Sparten wie Kultur, Sport oder aus der allgemeinen Dienstleistungsbranche.
Die Anbieter stellen dem Hotel Sonderkonditionen zur Verfügung, wie z. B. bei einer Autovermietung: Das Auto wird bei Mietbeginn dem Gast des Hotels vor Ort bereitgestellt und auch wieder abgeholt. Das Ziel der Anbieter ist, den Service gegenüber dem Hotel so zu verallgemeinern, dass man dieses Angebot auf weitere Ferienclubs ausweiten kann. Ebenso wie bei dem Ferienclubmuss für die Bereitstellung der Angebote eine Anwendung vorhanden sein.Dabei muss dieser Bereich in die bestehende Umgebung der Anbieter neu eingegliedert werden.

Gast

Der Gast nutzt die Angebote des Hotels. Er bekommt bei der Ankunft im Ferienclub ein mobiles Gerät ausgehändigt, mit dem er sich die einzelnen Leistungen anzeigen lassen kann und diese dann auch bucht. Er hat somit während seines Aufenthaltes Zugang zum Informationssystem des Ferienclubs und muss auch über die Vorgehensweise bei kostenpflichtigen Angeboten informiert werden. Bei der Abreise wird das Gerät wieder zurückgegeben und zurückgesetzt.
Der Gast benötigt ein mobiles Gerät mit Funkmodul oder alternativ Terminals, die im gesamten Hotelbereich verfügbar sind, damit es möglich ist, sich im Informationssystem des Hotels anzumelden. Über eine Anwendung auf dem PDA werden die Leistungen des Ferienclubs abgerufen und zur Verfügung gestellt. Die Anwendung dient auch zur Reservierung der Angebote.

3. Analyse

Um das geschilderte Szenario (siehe Kapitel 2) umzusetzen und die Möglichkeiten von Mono in dieser Umgebung zu evaluieren, sind vorher einige Anforderungen zu bedenken. Diese sollen später in dem Design des Systems mit einbezogen und im Prototypen auf Funktionsfähigkeit untersucht werden. Es müssen die Voraussetzungen der Anwendung selbst, die von dem Hotel und den Gästen beachtet, bzw. erfüllt werden. Weiterhin entstehen so Bedingungen, die sich auf das Umfeld und die technische Umsetzung auswirken. Von besonderem Interesse ist die Middleware, worüber die Anwendung mit den einzelnen Anbietern kommunizieren soll, um Angebote zu erhalten und Reservierungen vornehmen zu können.

Eine Anwendung soll auf dem PDA (Portable Digital Assistant) als Testfeld dienen. Es wird überprüft, inwieweit eine Middleware mit Web-Services auf Mono als dessen Plattform dienlich ist und die einzelnen Komponenten nutzbar sind. Es spielen dabei Begriffe, wie z. B. Skalierbarkeit, Portabilität oder Sicherheit ein Rolle. Mono sollte Möglichkeiten (Werkzeuge) bieten, die das Umsetzen der genannten Punkte bei einer Middleware, bzw. eine Anwendung, unterstützen.

Eine “zusätzliche” Überlegung ist die Portabilität der Anwendung auf ein anderes System. Durch Web-Services soll ermöglicht werden, heterogene Systeme miteinander kommunizieren zulassen. Ist es also möglich, Mono als Plattform und Web-Services als Technologie zu nutzen, um ein Zusammenspiel mit anderen Plattformen, wie z. B. JAVA, zu ermöglichen [Czarski 2005]?

Am Ende des Kapitels soll dargestellt werden, welche Anforderungen sich für den Einsatz von Mono als Plattform für Web-Services auf mobilen Endgeräten ergeben. Abschliessend werden verschiedene mögliche Geschäftsprozesse vorgestellt, die in die Anwendungsarchitektur einfliessen.

3.1. Rolle der Anwendung

Eine Anwendung, entwickelt aus dem Szenario und den Anforderungen, soll als Testfall für Mono dienen, um zu prüfen, wie gut Services in einer realen Umgebung nutzbar sind. Die Anwendung besteht aus mehreren Komponenten. Diese nutzen die Middleware um in einem

verteilten System, bestehend aus mobilen und stationären Geräten, zu interagieren. Die Interaktion zwischen den Komponenten wird mittels Web-Services realisiert, um die technologischen Anforderungen an Mono abzuschätzen.

Anforderungen aus der Anwendung

Aus Sicht des Gastes gilt:
Ein Gast wird das Angebot eines Drittanbieters nutzen, wenn er dadurch seinen Urlaubsaufenthalt angenehmer und flexibler gestalten kann, dass z. B. dadurch dass eine grössere Unabhängigkeit und Bewegungsfreiheit gewährt wird. Selbiges wird hier an dem Beispiel einer Autovermietung erklärt: Der Gast kann zu beliebiger Tages- und Nachtzeit über sein Handheld ein Auto bei einer Fremdfirma oder über das Hotel mieten. Dabei entfällt die Beachtung von Öffnungszeiten und die zusätzlich Suche nach einer Filiale oder einer Vermietungsstelle, was zusätzlich Zeit spart. Ein weiterer Vorteil ist die Einfachheit einer solchen Buchung. Es gibt keine lange Wartezeiten beim Reservieren und der Ablauf der Buchung ist “stressfreier”. Es ist möglich die Buchung direkt von dem Hotelzimmer aus vorzunehmen, oder beim Frühstück im Hotel oder vielleicht am Strand.

Aus Sicht des Hotels gilt:
Der Service muss “einfach funktionieren”. “Funktionieren” bedeutet in diesem Zusammenhang, dass der Gast keine Schwierigkeiten in der Benutzung hat und der Service robust ist. Das Hotel möchte nicht nur einzelne Profile von Gästen (z.B. Informatiker, Jugendliche) bedienen, sondern allen Gästen denselben Service zur Verfügung stellen. Ein wesentlicher Punkt in der Betrachtung ist: keine Kompromisse in der Bedienbarkeit des Gerätes. Der Gast ist im Urlaub und möchte sich erholen. Ein Gerät, welches ihm eher hinderlich als hilfreich ist, wird somit abgelehnt. Für das Hotel ist es eine zusätzliche Belastung durch die Aufrechterhaltung eines Services, der für den Urlauber uninteressant ist. Die Folge ist ein unzufriedener Gast.

Ein weiterer Aspekt aus der Sicht des Hotel ist: Der Dienst des Drittanbieters muss leicht in die IT-Landschaft des Hotels zu integrieren sein. Das Hotel darf nicht gezwungen sein, die eigene Umgebung auf jeden einzelnen Anbieter anzupassen oder, bei dessen Änderungen, die eigene Umgebung verändern zu müssen. Die Dienste müssen mit dem Hotel agieren ohne zusätzliche Anforderungen von Seiten der Fremdanbieter.

Ergonomie

Bietet das Hotel an, mit Hilfe eines PDA Reservierungen (z. B. Autovermietung, Stadtführung) vorzunehmen, sollte beachtet werden, dass der Gast kein Interesse an einer langwie-

rigen Einweisung in die Bedienung des Gerätes hat. Das mangelnde technische Verständnis des Gastes muss mit einbezogen werden. Das Gerät ist so zu konzipieren, dass die Technologie für jede Person ohne Verständnis derselben bedienbar ist. Die Oberfläche für die Anwendung muss so aufgebaut sein, dass das “look & feel” dem von Windows entspricht und dementsprechend zu bedienen ist. Damit ist die Wiedererkennung von bekannten Komponenten des eigenen PC-Systems gewährleistet und die Bedienung vereinfacht (z. B. keine Tastaturkürzel, die sich an einem Appelsystem orientieren).

Es sollten ebenfalls vom Benutzer keine Interaktionen verlangt werden, die nicht direkt für die Anwendung nötig sind. Dazu zählen z. B. wiederholte Passworteingaben um ein Netzwerkzugang zu erhalten oder Einstellungen für eine Bluetoothverbindung (vierstelliger PIN). Es soll möglichst ein “single sign on” geben. Der Gast meldet sich am Gerät einmal an und kann alle Angebote erreichen.

Mono muss auf den verschiedenen Betriebssystemplattformen eine einheitliche Bedienoberfläche zur Verfügung stellen können (graphische Bibilotheken) und mit dem Betriebssystem interagieren können.

Internationalisierung

Ein Hotel beherbergt meist Gäste verschiedener Nationalitäten, somit verschiedener Sprachen. Die Angebote sollten daran angepasst werden und in verschiedenen Sprachen verfasst sein. Die “rich Clients” sollten ebenfalls an die verschiedenen Nationalitäten angeglichen werden, um den Gästen den Umgang weiter zu erleichtern. Das Gerät muss auf die jeweilige Landessprache der Gäste einstellbar sein. Gebührenpflichtige Reservierungen oder Verträge (z.B. Mietwagenreservierungen) sind am besten in der Muttersprache zu verstehen. Durch das bessere Verständnis entsteht keine Unsicherheit darüber, ob die Reservierung auch die richtige war. Damit bleibt dem Benutzer das umständliche Buchen an der Rezeption oder dem direkten Reservieren am Schalter, vielleicht sogar in einer fremden Sprache, erspart. Fachbegriffe und unverständliche Abkürzungen dürfen beim Bedienen des Gerätes nicht auftreten. Die Anwendung muss einfach und verständlich sein. Das könnte z. B. heissen: Ein deutscher Gast fordert in Frankreich die Angebote über Mietwagen an. Die Angebote sind auf Grund der Landessprache auf Französisch. Das Hotel muss nun die Angebote für den Gast auf deutsch bereitstellen können. Die Software muss die Angebote abholen und bei sich für den Gast übersetzen bevor sie ihm diese zur Verfügung stellt.
Dieselbe Forderung ist bei der Reservierungsanforderung für den Rückweg gültig.

Es sollten neben den speziellen PDA Eigenschaften (z. B. Schrifterkennung) auch die Oberfläche und die Bedienung der Anwendung an die jeweilige Sprache angepasst sein. Ein möglicher Zusatz auf dem Gerät wäre, die Preise für ein Angebot in der Währung des Gastes anzuzeigen. Der Gast hätte dann ein besseres Verständnis für das Preis-Leistungsverhältniss.

Bei der Buchung könnte der Preis durch die Anwendung in die Landeswährung umgerechnet werden.

Weiterhin muss die Datumsdarstellung beachtet werden, die beispielsweise Unterschiede aufweist zwischen Deutschland und den USA. Die richtige Dezimaldarstellung der Rechnungsbeträge ist eine weitere wichtige Anforderung. Es muss darauf geachtet werden, wie die einzelnen Gäste Punkt oder Komma interpretieren. Zu den aufgezählten Punkten kommt weiterhin noch die Darstellung einer Woche hinzu. In manchen Ländern beginnt die Woche am Sonntag oder am Montag.

Es muss die Möglichkeiten geben sein eine Internationalisierung vornehmen zu können. Mono sollte eine Bibliothek besitzen, mit der es möglich ist, die Anwendung in verschiedenen Sprachen und Währungen darzustellen.

Robustheit

In einem Hotel kann es immer wieder einmal zu Störungen im Betrieb kommen oder zu Engpässen bei bestimmten Serviceleistungen, die bei den Gästen besonders beliebt sind oder wo die Organisation einen höheren Aufwand erfordert.

Ähnliche Probleme können bei der Benutzung von Anwendungen auftreten. Bei Geräten die mit Funkmodulen ausgestattet sind, kommt es sehr häufig zu Verbindungsabbrüchen. Dies ist besonders hinderlich, wenn der Urlauber eine Reservierung vornehmen möchte oder mit einem Bekannten über das Gerät kommuniziert. Ist der Gast nur dann in der Lage das Gerät zu benutzen, wenn er sich innerhalb des Hotelnetzes mit sehr gutem Empfang befindet, ist es für ihn nicht von Vorteil und er kann darauf verzichten. Ein PDA soll dem Nutzer eine Möglichkeit geben mit dem Hotel oder deren Dienstanbietern zu kommunizieren, wenn er sich bedingt im Umfeld der Ferienanlage befindet. Der Gast sollte nicht gezwungen sein, eine Reservierungsanfrage mehrmals aufgeben zu müssen. Die Gründe dafür können sein: eine Verbindung zum “Hotelserver” kommt nicht zustande oder bricht während der Übertragung ab. Zusätzlich ist eine Regelung erforderlich, die eine Synchronisation der Angebote ermöglicht.

Jeder Computerbenutzer kennt die Probleme, wenn der Bildschirm “einfriert” und nichts mehr funktioniert. Man ist in diesen Momenten gezwungen den Rechner neu zu starten und manchmal auch Arbeitsabläufe zu wiederholen, weil durch den Absturz des Computers die Daten verloren gegangen sind. Die Hotelanwendung muss robust gegenüber solchen Ausfällen sein. Wenn Fehler während des Betriebes auftauchen, sollte das Gerät und die Software die Möglichkeit haben, sich in einen Zustand “zurückzusetzen”, der dafür sorgt, dass die Software oder das Gerät sich nicht “aufhängt” und es weiterhin bedienbar bleibt. Der Gast muss die Möglichkeit haben, sich zumindest wieder abzumelden oder persönliche

Daten zu löschen. Es dürfen keine Informationen im Speicher des Gerätes bleiben, um dritten Personen Zugriff darauf zu geben. Es ist denkbar, dass Personen durch ein absichtliches Herbeiführen dieser Situation die Sicherheitsfunktionen des Gerätes umgehen wollen. Diese würden sich somit unberechtigter Weise Zugriff auf das System oder Zugriff auf das Netz verschaffen (Die Frage der Sicherheit dafür wird ausführlicher im Kapitel 3.2 behandelt.).

Die beschriebenen Probleme betreffen nicht nur die Anwendung, sondern zusätzlich das Betriebssystem über das die Speicherverwaltung, die Netzwerkverbindungen sowie die Authentifizierung abläuft.
Mono muss für die Anwendungsprogrammierung über ein Konzept zur Fehlerbehandlung verfügen. Mit der Fehlerbehandlung sollte es möglich sein, den Benutzer über den gemachten oder entstandenen Fehler “sinnvoll” zu informieren. Es sollte die Anwendung in einem Zustand halten, der eine weitere Bedienung trotz Fehlers möglich macht. Mono selbst muss als Plattform in einer stabilen Version erhältlich sein und darf nicht nachvollziehbar oder sporadisch im Betrieb ausfallen.

Datensicherung, Synchronisation

Wenn im oberen Kapitel die Forderung nach Robustheit der Anwendung auftrat, sollte weiterhin überlegt werden, was getan werden kann, wenn die Anwendung trotz der Vorsichtsmassnahmen abstürzt.

Die Anfragen eines Gastes könnte auf dem Gerät bei jedem Schritt zwischengespeichert werden solange diese in Bearbeitung sind. Somit ist gewährleistet, dass bei Problemen mit der gewählten Reservierung immer nachvollziehbar bleibt, welchen Status die Reservierungsanfrage besitzt (z. B. nicht bestätigte Reservierung oder fehlerhafte Abrechnung). Bei Verbindungsproblemen könnte die Middleware die Daten wiederholt vom PDA bekommen, ohne den Gast zur Wiederholung der Reservierung auffordern.

Portierbarkeit

Findet in dem Hotel oder der Hotelkette dieser Service einen guten Anklang bei den Gästen, kann man weitere Leistungen hinzufügen. Durch die schnelle Weiterentwicklung der Techniken und der steigenden Performance auf mobilen Geräten (z. B. PDA, Mobiltelefon) sollte eine Lösung gefunden werden, mit der die Portierbarkeit auf verschiedene Geräte möglich ist. Auf Anwendungsebene ist darauf zu achten, dass das System möglichst modular entwickelt ist, um später einzelne Komponenten (z. B. GUI) auszutauschen oder zu erweitern. Es ist denkbar im Hotel Terminals aufzustellen, die den Gästen den selben Service zur Verfügung stellen wie der PDA und somit die gleiche Anwendung nutzen.

Bei der Portierung auf verschiedene Systeme muss daher gesichert sein, dass Mono dort nutzbar ist. Eine Überlegung in diesem Zusammenhang kann sein, dass man Mono auf Windowsbasierten System mit .NET ersetzt.

3.2. Konsequenzen für die Middleware

In den oberen Kapiteln wurde beschrieben, was die Erwartungen an die Anwendungen sind um erfolgreich in einem Ferienclub zu bestehen. Es wurden mehrere Anforderungen formuliert, die zu verschiedenen Bedingungen für die Middleware führen. Die Middleware ist die Kommunikationskomponente für die Interaktion mit den verschiedenen Hoteldiensten und weiteren Anbietern. Es muss möglich sein darüber Daten auszutauschen, Angebote abzufragen, oder verbindliche kostenpflichtige Dienste, Reservierungen (z. B. Autovermietung), zu benutzen. Dieses muss unabhängig davon sein, was für eine Art Angebot es ist oder wer es anbietet. Die Uhrzeit spielt ebensowenig eine Rolle, wie wer es bucht, solange es ein Gast des Ferienclubs ist (keine Abstimmung, wie z. B. dieser Dienst benötigt die Zustimmung des Hotelpersonales). Siehe hierzu auch die Anforderungen aus der Anwendung (Kapitel 3.1). Die Forderungen des Gastes müssen zwischen den Servicen bei der Kommunikation Berücksichtigung finden.

Wichtige Merkmale sollten die Sicherheit und der Ablauf einer Buchung sein. In dem Hotel werden Leistungen angeboten, wie Autovermietung oder Tauchkurse, die viel Geld kosten können. Es muss gesichert sein, dass die Middleware eine sichere Transaktion gewährleistet. Auf Grund der kostenpflichtigen Dienste muss klar definiert sein, wie der Ablauf einer Buchung ist und was im Falle von Problemen bei den Kommunikationspartnern abläuft.

Die eingesetzte Middleware des Hotels muss mit den verschiedenen Plattformen der Anbieter kommunizieren können. Die Anbieter können bei ihrer internen Struktur nicht gezwungen werden die gleiche Plattform zu verwenden wie der Ferienclub. Es muss eine plattformunabhängige Kommunikation stattfinden, die auf der bestehenden Infrastruktur aufbauen kann.

Ergonomie

Es gibt bestimmte Anforderung, was die Ergonomie der obigen Anwendung betrifft. In der Middleware spielen diese Anforderungen nicht direkt eine Rolle. Es muss aber darauf hingewiesen werden, dass es bei Fehler in der Ausführung einer Reservierung über den Zugriff durch die Middleware keine Meldungen geben darf, die an den Benutzer weitergegeben werden dürfen. Die Meldungen müssen an dass dafür zuständige Personal gemeldet werden und der Gast bekommt bzgl. seiner Aktion ein Meldung die nur aussagt, dass dieser Service leider nicht zur Verfügung steht. Ein Beispiel wäre: Der Gast darf nicht von technischen

Problemen, wenn diese auftreten, belästigt werden oder “kompliziert technische” Meldungen erhalten, bzw. ebensolche Fehlermeldungen bekommen. Die einzigen Meldungen die der Gast erhält sollten seine Reservierungs- und Angebotsbenachrichtigungen sein.

Daraus ergibt sich für Mono die Forderung nach einem Konzept zur Fehlerbehandlung.

Internationalisierung

Das Hotel kann zu einer weltweiten Kette von Hotels gehören. Somit muss die Kommunikation zwischen der Hauptniederlassung und den Hotels gewährleistet sein. Es müssen Schnittstellen geschaffen werden, die es ermöglichen, dass sich die Mitarbeiter verständigen können, egal in welchen Land diese arbeiten. In diesem Zusammenhang ist daher die Datumsdarstellung wichtig für die Koordination. Angaben über die Zeitzonen oder, wie oben bereits erwähnt, die Darstellung wann eine Woche beginnt, müssen beachtet werden. Genauso muss die Darstellung der richtigen Währung gewährleistet sein, Beispiele dafür können der Euro oder der amerikanische Dollar sein. Bei einem internationalen Unternehmen, wie einer Hotelkette, ist zusätzlich die sprachabhängige Gestaltung von Menüs, Nachrichten, etc. wichtig. Die Middleware sollte mit Hilfe von Mono Mechanismen für den Umgang mit verschiedenen Zeichensätzen, Währungen oder Kalenderdaten bereitstellen.

Robustheit

In dem Ferienclub werden mobile Endgeräte mit Funkmodulen eingesetzt. Eine Eigenschaft dieser Geräte ist, dass die Verbindungen zu den Servern als nicht sehr stabil einzustufen sind. Die Gäste werden die PDAs spontan im Netz anmelden und wieder abmelden. Eine Vermutung ist, dass die Gäste sich nicht immer an den vorgeschriebenen Ablauf halten werden (das Abmelden aus dem Netzwerk). Für diese Fälle muss sichergestellt sein, dass das Gesamtsystem, bzw. die Middleware, mit allen anderen aktiven Geräten im System weiter funktioniert. Der Ausfall eines Gerätes im Netz darf keinen Einfluss auf den restlichen Betrieb zur Folge haben. Ein Beispiel hierfür wären die vielen verschiedenen Prozesse (Threads), die parallel ablaufen. Diese können zur Beeinträchtigung des Systems führen, wenn der Prozess beim Abbruch des Vorgangs nicht beendet wird. Diese Probleme müssen bei der Implementierung berücksichtigt werden. Mono sollte Konzepte für die Nebenläufigkeit und Fehlerbehandlung bei der Entwicklung zur Verfügung stellen.

Cachemechanismen/Datenreplikation

In der Hochsaison kann es aufgrund der hohen Auslastung der Hotelbelegung immer wieder zu Engpässen in den Netzwerken des Hotels und der Anbieter kommen. Zur Entlastung des Netzwerks können populäre Angebote von Fremdanbietern direkt beim Hotel abrufbar sein. Den Gästen wird somit ein besserer Service geboten. Bei der Schilderung des Szenarios wurde darauf hingewiesen, dass es zwei Möglichkeiten geben kann, dem Gast die Angebote zur Verfügung zu stellen. Eine dieser Möglichkeiten wäre den Hotelserver als Proxy zu benutzen. Hierbei müssen Mechanismen entwickelt werden, die die Angebote synchron halten. Ein Gast könnte z. B. ein Angebot anfragen, das nicht auf dem Hotelserver erhältlich ist (bei der Autovermietung, ein bestimmter Wagentyp). Das Hotel leitet die Anfrage weiter und hält die Informationen für weitere Anfragen lokal vor. Es muss entscheiden, welche Angebote längere Zeit vorgehalten werden, wie oft sie akualisiert werden sollen oder wann sie gelöscht werden. Diese Überlegungen sind technischer Art und beinhalten keinen direkten fachlichen Bezug zur Umsetzung von Reservierungsabläufen oder das Senden von Angeboten. Es wäre wünschenswert, dass Mono eine organisierte Umgebung für die Infrastrukturdienste zur Verfügung stellt, so wie es von den Applikationsservern aus dem J2EE Umfeld her bekannt ist.

Erweiterbarkeit, Offenheit

Nach der ersten Inbetriebnahme des Systems und den ersten Buchungen wird sich herausstellen, inwiefern dieser Service rentabel ist ob die Gäste zufriedener sind. Sollte sich dann herausstellen, dass das System eine breite Akzeptanz findet, entsteht von Seiten des Hotels der Wunsch nach einer Vergrösserung des Angebots. Es werden mehr Schnittstellen zu anderen Anbietern benötigt und intern muss die Serviceleistung auf diesem Bereich verbessert und vergrössert werden. Somit muss die Kommunikationsplattform so aufgebaut sein, dass diese an weitere Anbieter angepasst werden kann und möglichst über eine Zugriffstransparenz verfügt, die es ermöglicht alle möglichen Serviceleistungen einfach zu integrieren. Die Middleware muss weiterhin mit den verschiedenen Plattformen der Fremdanbieter interagieren können. Es sollte eine möglichst unabhängige Schicht sein die einen Datenaustausch ermöglicht, aber keine Abhängigkeit zu den Geschäftslogiken der einzelnen Teilnehmer besitzt. Ändert ein Anbieter seine Plattform und seine Software, muss die Middleware weiterhin als Kommunikationskomponente dienen, die den Anbieter nach aussen zu dem Hotel präsentiert.

Mono muss bei der Nutzung von der Middleware die Spezifikationen der eingesetzten Techniken einhalten, so daß es mit anderen Systemen interagieren kann. Die Einhaltungen von Standards ist ebenfalls erforderlich für die mögliche interne Erweiterung mit anderen Technologien.

Transaktion

Die korrekt funktionierende Anwendung auf den mobilen Geräten ist ein wichtiger Anteil, um dem Gast die korrekte Reservierung von Angeboten zu gewährleisten. Um aber die Reservierung des Gastes abzuwickeln sind weitere Punkte erforderlich, die hier erläutert werden sollen.

Bei der Reservierung durch das Hotel bei einem Anbieter handelt es sich um einen Geschäftsprozess der rechtlich bindend ist und eine finanzielle Abrechnung nach sich zieht. Dieser Teil der Buchung muss als Transaktion durchgeführt werden, die garantieren soll, dass sich die Reservierungen immer in einem konsistenten Zustand befinden.

Es gibt zwei Arten von Transaktionen: Die verteilte Transaktion, bei der mehrere von einander getrennte Komponenten daran teilnehmen. Ein Beispiel wäre die Reservierung eines Autos. Die Autovermietung bietet die Leistung und organisiert den Ablauf. Das Hotel übernimmt die Reservierung und die Abrechnung mit dem Gast. Sollte sich jetzt einer der Punkte ändern, so hat dies Auswirkungen auf alle Beteiligten. Es muss dafür gesorgt werden, dass alle Teilnehmer über mögliche Zustandsänderungen informiert werden. Bei der atomaren Transaktion gibt es nur zwei Beteiligte. Diese Transaktion ist übersichtlicher und einfacher zu behandeln. Beim weiteren Verlauf der Untersuchung wird von atomaren Transaktionen ausgegangen. Verteilte Transaktionen werden wegen ihrer Komplexität vernachlässigt, weil es nicht Gegenstand dieser Arbeit ist.

Bei der Transaktion sollen bestimmte Vor- und Nachbedingungen erfüllt sein. Es wird sich bei dem Beispiel der Autovermietung an den Bedingungen ähnlich denen eines Vertrages orientiert. Der Gast muss, als Vorbedingung, einen Führerschein besitzen, Gast des Hotels sein, die Dauer der Vermietung muss im Zeitraum des Aufenthaltes liegen und es können nur die vorliegenden Angebote genutzt werden. Nach der Buchung ist das Hotel verpflichtet dem Gast einen Wagen zur Verfügung zu stellen. Das Beispiel der Autovermietung wird “vereinfacht” indem das Hotel vorher prüft, ob alle Daten korrekt sind.

Was passiert mit einer Reservierung, die durch ein Verbindungsabruch oder ähnliches in dem Ablauf unterbrochen wird? Es gibt mehrere Seiten zu betrachten: An welcher Stelle wird der Vorgang abgebrochen, in welcher Situation und wer erkennt den Abruch (Server oder Client)? Die nachfolgende Tabelle 3.1 soll einen kurzen Überblik geben und stichwortartig zeigen, wie darauf reagiert wird.

- Der Hotelserver (Client), der die Buchung durchführt, bricht die Verbindung ab. Dann muss der Fremdanbieter (Server) die angefangene Reservierung löschen und eine Meldung über die fehlerhafte Buchung herausgeben. Sollte der Hotelserver erkennen, dass der Fremdanbieter die Anfrage nicht abarbeitet (z. B. in einem bestimmten

	Client	Server	Netzwerk	Aktion
Status aus Sicht des Clients	✓	✓	✓	Die Reservierung ist erfolgreich
Die Verbindung bricht ab, Timeout	✓	✓	↯	Abbruch der Transaktion, Rückmeldung an den Anwender
Erkennung des Absturzes eines Server-Prozesses	✓	↯	✓	Transaktion nicht mehr gültig, Rückmeldung an den Anwender
Der Client selber stürzt ab	↯	✓	✓	Die Operation kann nicht durchgeführt werden
Status aus der Sicht des Servers	✓	✓	✓	Die Reservierung ist erfolgreich
Das Netzwerk ist nicht mehr verfügbar	✓	✓	↯	Alle Transaktionen werden abgebrochen
Der Server-Prozess stürzt ab	✓	↯	✓	Ein neuer Prozess bricht alle Transaktionen ab und benachrichtigt den Client.
Der Client ist nicht mehr erreichbar	↯	✓	✓	Wenn in einer vorgegebenen Zeit die Transaktion nicht abgeschlossen ist, wird diese abgebrochen.

Tabelle 3.1.: Mögliche Transaktionsfehler

Zeitraum antwortet), muss der Client die Anfrage abbrechen. Es muss eine Benachrichtigung gesendet werden oder die Daten werden ein weiteres Mal versendet. Eine erneute Versendung kann zu dem Problem führen, dass der Fremdanbieter die erste Anfrage noch im System hält und dadurch Duplikate entstehen.

- Die Buchungsanfrage beim Fremdanbieter wird abgebrochen. Ein neu gestarteter Prozess muss die nicht erfolgreiche Transaktion abbrechen und die begonnene Transaktion zurücksetzen. Clients, die während einer Transaktion abbrechen, wird ein Zeitraum gesetzt in dem sie antworten müssen. Nach Ablauf der Zeit wird die Transaktion abgebrochen. Danach erfolgt eine Mitteilung, dass die Reservierung nicht erfolgreich war.

Für die Nutzung von Transaktionen gibt es für die Web-Services verschiedene Konzepte [Cabrera 2004a], [Cabrera 2004b]und weiter entwickelt werden. IBM und Microsoft haben ein neues Konzept auf Basis dieser Standards entwickelt, die der Vereinfachung von Transaktionen dienen sollen [Little und Freund 2003]. Microsoft hat diese Entwicklung in .NET mit eingebunden. In diesem Zusammenhang sollte Mono diese Vorgaben ebenfalls erfüllen und das Framework dazu bieten.

Sessionmanagement, Multiuser

Durch die Größe eines Ferienclub ist davon auszugehen, dass es eine Vielzahl von Benutzern in diesen Systemen gibt. Daraus ergeben sich viele verschiedene Probleme was die Bearbeitung der einzelnen Reservierungen angeht.

Das System muss als Mehrbenutzersystem ausgelegt sein um mehrere Benutzer im System zu halten. Die Abarbeitung der einzelnen Reservierungen muss unabhängig von anderen gleichzeitig ausgeführten Reservierungen stattfinden. Der jeweilige Vorgang darf nicht von anderen abhängig sein. Wenn nun zwei Gäste parallel dasselbe Angebot buchen und z. B. bei einer Stadtführung nur noch ein Platz zu vergeben ist, gilt als erfolgreiche Buchung jeweils diejenige, die vom Hotel als erstes abgerechnet wurde.

Wenn Reservierungen bei Anbietern durchgeführt werden, dürfen diese nicht von parallel laufenden Anfragen behindert werden. Die Operationen müssen erfolgreich abgeschlossen werden oder dürfen keine Wirkung haben [George Coulouris 2002]. Es handelt sich um eine dauerhafte Änderung. Die Reservierung ist gebucht und hinterlegt. Sie kann auch bei Absturz des Systems wieder geholt werden (Kapitel 3.2).

Diese Punkte sind die Voraussetzungen für einen reibungslosen Ablauf der Reservierungen in dem Hotel. Aber was muss geschehen, wenn es zu Verbindungsabbrüchen kommt oder zu Schwierigkeiten im Netzwerk, sei es das Hotelnetz oder die Verbindung zum Fremdanbieter?

Es ist möglich, dass der Gast mittels einer Funkverbindung in seinem Hotelzimmer eine Reservierungsanfrage stellt und nach der erfolgreich abgesendeten Anfrage sein Zimmer verlässt, die Verbindung unterbricht und sein PDA erst in der Hotelhalle wieder eine Verbindung zum Hotelnetz erhält, wobei es diesmal eine Wlan-Verbindung ist. Somit müssen die verschiedenen Services als Information mitgesendet bekommen, wer eine Anfrage gestellt hat, um diese später wieder herstellen zu können. Der Server muss wissen, welcher mobile Client die Anfrage stellt und wohin er die Antwort liefern muss falls der Client kurzzeitig die Verbindung verliert und nicht erreichbar ist.

Sinnvoll ist es nun die Anfrage asynchron zu stellen. Aus dem Grund, dass die Übertragung der Informationen über das Netz länger dauern kann (z. B. Bluetooth) oder auch, weil es wie oben zu Verbindungsabbrüchen kommen kann. Der Client kann mit weiteren Aufgaben weiterarbeiten und die Web-Services (WS-Proxy) warten auf die Antwort, die bei Erhalt weiter gereicht wird.

Da der Gast seine Wünsche (Anfragen und Reservierungen) nur mitteilt aber nicht direkt selber vornimmt, sind Verbindungsabbrüche während der Kommunikation zum Hotelserver zu beachten, haben aber keine massive Einwirkung auf den Buchungsablauf. Der Server hält immer den aktuellen Stand der Kommunikation zu dem Gast fest und kann somit die Kommunikation an unterbrochener Stelle fortführen oder es ist auch denkbar, dass Gäste direkt über Terminals im Hotel sich die Informationen geben lassen (siehe Diplomarbeit [Bresch 2004]).

Sicherheit

Ein wichtiger Aspekt in der Anforderung ist die Sicherheit. Ohne Gewährleistung dieser kann solch eine Dienstleistung nicht angeboten werden. Es sind viele verschiedene Punkte zu beachten, die Andre Lüpke [Lüpke 2004] in seiner Diplomarbeit genauer ausgeführt hat.

An dieser Stelle möchte ich kurz auf die Punkte eingehen, die bei der Umsetzung von dieser Umgebung beachtet werden müssen.

Der Gast muss sicher sein können, dass seine Daten geschützt und vertraulich behandelt werden. Es müssen alle datenrechtlichen Richtlinien eingehalten werden. Bei der Übertragung von personenbezogenen Informationen, wie Kreditkartennummern ist höchste Sicherheit zu garantieren.

In einem ersten Punkt ist es wichtig, dass ein Hotel oder Ferienclub die geltenden Datenschutzgesetze einhält. Der Ferienclub muss die Gäste darüber informiert, was mit ihren Daten geschieht. Transparenz sorgt für ein Gefühl der Datensicherheit. Mit Transparenz ist gemeint, dass der Gast darüber informiert werden kann, wenn dieser es wünscht, wo und wie seine Daten geschützt werden.

Beim Austausch der Daten über eine Funkverbindung (in diesem Fall PDA und Hotel) sollten diese verschlüsselt werden. Funknetze werden bisher eher als unsicher einzuschätzen, was das Abhören von Nachrichten betrifft. Somit muss sichergestellt werden, dass beim “sniffen“ die Daten nicht lesbar sind. Zusätzlich benutzt die Middleware als Netzwerkverbindung zwischen dem Hotel und Anbietern das Internet, welches in diesem Umfeld nicht als sicher einzustufen ist, weil dort Hoteldaten übertragen werden, die nicht öffentlich gemacht werden dürfen. Ebenfalls müssen die Datenbestände in dem Hotelnetz, die Anwendungsprogramme und das Netz selber geschützt werden. Zu den zu sichernden Daten und den personenbezogen Informationen der Gäste, muss ebenfalls das System als solches geschützt werden. Das Netz muss ausfallsicher sein. Es darf nicht gezielten Angriffen wie z. B. „Denial of Service“ [1] zum Opfer fallen. Eine Folge dieser Angriffe kann den Ausfall des Netzwerkes bedeuten und somit Abbruch oder Misslingen von Transaktionen der Gäste. Die Buchungen von Veranstaltung können sich durch diese Art von Angriff in einem nicht definierten Zustand befinden (denkbar wäre indiesen Fall z. B. eine Überbuchung von Veranstaltungen). Die Daten der Gäste wären vor unbefugtem Zugriff nicht mehr sicher. Dabei muss ein Angriff gegen das Netzwerk nicht darauf abzielen es zum Erliegen zu bringen, sondern das Netzwerk zu manipulieren um daraus einen persönlichen Nutzen zu erhalten. Es müssen sämtliche Schutzmassnahmen (z. B. Firewalls, VPNs, etc. . . .) genutzt werden, um Hackern und Crackern den Zugriff auf das Netzwerk zu verwehren. Die Schwierigkeit liegt dabei in der Nutzung von Funk- und Bluetoothnetzwerken, die durch ihre offene Struktur ein grundsätzliches Sicherheitsloch darstellen, welches mit den oben genannten Methoden gesichert werden muss.

In einigen Hotels werden Funknetze angeboten [hotel.de 2003], mit denen man seine eigenen mobilen Geräte nutzen kann um ins Internet zu gelangen. In diesem Ferienclub dürfen nur PDAs aus dem Ferienclub genutzt werden. Die Nutzung der privaten Handhelds der Urlauber sollte untersagt bleiben. Da nicht garantiert werden kann, dass die Benutzung fremder Geräte im Hotel nicht zu Sicherheitsproblemen führt. Bei der Nutzung von privaten Geräten im Netzwerk ist nicht geklärt, welche Sicherheitsmassnahmen auf dem Gerät ergriffen wurden (z. B. Sicherheitsupdates, Firewall). Dem Ferienclub ist nicht bekannt, wie die Gäste ihre eigenen Geräte nutzen. Gäste könnten bei Fehlern auf ihrem PDA das Hotel dafür verantwortlich machen oder andere missbrauchen das Netzwerk. Ein weiterer Grund des Verbots ist der zeitliche Aufwand, den Gästen auf ihren Geräten die Einstellungen und die Anwendung aufzuspielen.

Die Middleware muss Schutzmechanismen besitzen für die genannten Punkte. Mono sollte dabei diese unterstützen und diese mit eigenen Konzepten für die Implementierung erweitern.

[1] DoS (Denial of Service) oder DDoS (Distributed Denial of Service) sind Angriffe auf Server mit dem Ziel sie und ihre Dienste arbeitsunfähig zu machen.Wiki.dos:2004 [2004]

Skalierbarkeit

Das System sollte höchst performant sein, denn durch langes Warten würde der Gast verärgert werden. Es sollte die jeweiligen Anfragen so schnell wie möglich durchführen, um den Gast bei der Planung seiner Urlaubsaktivitäten keine umständliche Terminplanung aufzuerlegen. Für das Hotel bedeutet dies, genügend Kapazitäten zur Verfügung zu stellen, um diesen Wunsch gewährleisten zu können. Dabei ist es für das Hotel wünschenswert, dieses von Computer umsetzen zu lassen, um das Personal im täglichen Umgang mit den Urlaubern zu entlasten.

Es ist aus der Sicht des Gastes wichtig, eine schnelle, einfache und positive Antwort auf seine Anfrage zu erhalten. Für das Hotel ist es wichtig, dass es nicht zu Fehlern bei Reservierungen kommt oder fehlerhaften Aussagen über das Angebot. Das Hotel möchte aus diesen Gründen mehr als nur eine Überprüfung und Buchung des Angebotes.

Dabei ist besondere Beachtung darauf zu legen, dass in der Hochsaison Engpässe in der Buchung passieren und dass man die höhere Auslastung mitberücksichtigen muss. Die gesamte Struktur muss darauf ausgerichtet sein. Dabei ist es zum einen entscheidend, dass das Hotel genügend Ressourcen besitzt, aber auch die Fremdanbieter ihr System genügend groß gewählt haben, um allen Verpflichtungen gegenüber ihren Kunden nachzukommen. Dabei könnte es z. B. so sein, dass eine Autovermietung mehrere Ferienclubs in der Umgebung als Kunden besitzt. In diesem Beispiel sollten dann die Web-Services, die als Schnittstelle dienen, eine zugesicherte Geschwindigkeit bieten die auch bei hoher Auslastung gewährleistet wird. Diese sind nur dann einzusetzen, wenn diese benötigt werden, z. B. bei der Buchung eines Fahrzeuges. Aus dem Anwendungsdesign ergeben sich die Forderungen nach einer guten Austauschbarkeit der Ressourcen. Wenn mit den Web-Services eine einheitliche Schnittstelle nach aussen angeboten wird, wo nur die angeforderten Daten übertragen werden und keine Abhängigkeiten zu der internen Verarbeitung bestehen, kann man die einzelne Ressourcen erweitern oder gegebenenfalls austauschen [skale:2004 2004]. Somit gilt: Es sollte ein “offenes System” sein, wo die Schlüsselschnittstellen der Komponenten einheitlich sind. Erreicht wird dadurch Flexibilität und Dynamik, um das System durch zukünftige Entwicklungen, Dienste und Komponenten einfach zu erweitern und deren Leistungsfähigkeit zu verbessern. Dabei muss betrachtet werden, dass die Komponenten für Web-Service, wie z. B. der Webserver über Möglichkeiten verfügt, mehrere “Monoprozesse” zu verwalten, um die Forderungen nach Verbesserung der Leistungsfähigkeit zu gewährleisten. Bei möglichen Datenbanken zur Verwaltung von Gästedaten sollte Mono auch mit mehreren Datenbanken umgehen können. Viele Aspekte können durch eine entsprechende Implementierung erreicht werden. Mono sollte dafür Möglichkeiten der Unterstützung dieser Implementierung anbieten.

Ein weiterer Punkt, der nicht direkt die Middleware betrifft, ist die Verbesserung der Band-

breite durch Aufstockung der Accesspoints auf der Anlage, die gewährleisten, dass die Verbindungen vom PDA und Hotelserver nicht zusammenbrechen.

3.3. Anforderungen an Mono

In den vorherigen Kapiteln wurden verschiedene Problem beschrieben, die von der jeweiligen Plattform gelöst werden sollten. Daraus ergeben sich jetzt verschiedenste Ansprüche, denen Mono nachkommen sollte um eine attraktive Umgebung zu sein.

- Mono muss eine Komponente zur Verfügung stellen, um Daten sicher gegen Unbefugte zu verschlüsseln, die sich z. B. durch “ sniffer” Angriffe Zugriff auf die Daten beschaffen wollen. Mono als Plattform muss somit bekannte und standardisierte Sicherheitstechniken unterstützen und zusätzlich einfach der Umgebung anpassbar sein, um veränderten Sicherheitsanforderungen schnell und flexibel umzusetzen. Dabei sollte es keine Rolle spielen auf was für einem System die jeweiligen Anwendungen genutzt werden.
- Die Web-Services unter Mono müssen eine gute Performance erhalten um den Anspruch einer schnellen Reservierungsbearbeitung zu gewährleisten. Da Web-Services grundsätzlich langsamer sind als andere Technologien (z. B. Corba) muss dort getestet werden, wie gut dieses Framework und Soap als Middleware unter Mono implementiert ist.
- Die Middleware muss in Netzen mit Funk- und Lanverbindungen einsetzbar sein, zusätzlich dabei auf verschieden Geräten wie z. B. PDA und Terminal.
- Ein Mehrbenutzerbetrieb, Nebenläufigkeit muss möglich sein. Es sollten mehrere Gäste auf einen Dienst gleichzeitig zugreifen können.
- Zur Fehlerbehandlung sollte Mono zudem ein Konzept bieten.
- Damit die Middleware im Hotel bei steigender Gästezahl weiter performant und ausfallsicher arbeitet, sollte es möglich sein diese skalierbar zu halten.
- Es muss die Möglichkeit geben eine Internationalisierung vornehmen zu können. Mono sollte eine Bibliothek besitzen mit der es möglich ist, die Anwendung in verschiedenen Sprachen und Währungen darzustellen.
- Mono muss auf den verschiedenen Betriebssystemplattformen eine einheitliche Bedienoberfläche zur Verfügung stellen können (graphische Bibliotheken).
- Wünschenswert wären Werkzeuge unter Mono, die eine organisierte Umgebung für die Infrastrukturdienste zur Verfügung stellen können.

3.4. Mögliche Geschäftsprozesse

Wie soll nun die Organisation der Reservierung aussehen? Wie kann der Gast auf die Angebote des Hotels und der anderen Anbieter, die mit dem Hotel zusammenarbeiten, zugreifen. Wie sollen die Angebote erfragt, in Anspruch genommen und im Verlauf des Urlaubs aktualisiert werden? Hier sind beispielhaft einige Abläufe vorgestellt, die eine Möglichkeit des Ablaufes bieten. Denkbar in diesem Zusammenhang ist auch die Annahme, dass alle Geschäftsmodelle genutzt werden um Angebote anzubieten. Es kann sein, dass die unterschiedlichen Anbieter unterschiedliche Vereinbarungen mit dem Hotel haben, wie reserviert und abgerechnet wird. Die verschiedenen Möglichkeiten, wie Angebote gebucht werden sollen zeigen, wie vielfältig die Prozesse sein können. Diese verschiedenen Möglichkeiten werden im weiteren erläutert und miteinander verglichen.

Die verschiedenen Buchungsmöglichkeiten:

1. Das Hotel sammelt alle Angebote und bietet diese den Gästen an. Der Gast bucht dann direkt beim Anbieter und bezahlt sofort.
2. Das Hotel sammelt alle Angebote und bietet diese den Gästen an. Der Gast bucht das Angebot beim Anbieter und dieser schickt die Rechnung an das Hotel, das dann die Abrechnung mit dem Gast vornimmt.
3. Das Hotel sammelt alle Angebote und bietet diese den Gästen an. Der Gast bucht beim Fremdanbieter und gleichzeitig wird die Rechnung erstellt und auf die Zimmerrechnung addiert.
4. Das Hotel sammelt alle Angebote und bietet diese den Gästen an. Dabei übernimmt das Hotel die Reservierung des Angebotes und die Abrechnung mit dem Gast und des Fremdanbieters.

Im ersten Schritt ist es bei allen Abläufen so, dass das Hotel die Angebote sammelt und den Gästen komplett zur Verfügung stellt. Somit kann das Hotel besser kontrollieren, welche Angebote dem Gast zur Verfügung gestellt werden. Es kann z. B. ein Anbieter den Service gegenüber dem Gast verschlechtert haben, in diesem Falle kann das Hotel kurzfristig das Angebot von der Liste der Angebote streichen. Die Reservierung und Abrechnung durch das Hotel durchführen zu lassen ist am sinnvollsten. Das Hotel setzt die Rechnung für in Anspruch genommene Dienste auf eine Gesamtrechnung, die bei Abreise bezahlt wird. Der Gast hat über das Hotel eine Kostenkontrolle und die Abrechnung erfolgt unkompliziert bei der Abreise. Ein weiterer Vorteil ist, dass der Gast bei Problemen direkt angesprochen werden kann und nicht über eine umständliche Kommunikation von Anbieter und Hotel der Gast erreicht wird. Zusätzlich kann dieses auch zu Sprachproblemen führen (siehe Internationalisierung). Der Gast wird zusätzlich ein besseres Gefühl haben, wenn er alle Leistungen aus

einer Hand bekommt und sich nicht mit den verschiedenen Anbietern auseinander setzen muss. Er hat in dem Hotel einen Ansprechpartner der ihm die gesamte Reservierung und Organisation abnimmt.

Aufgrund der hier erklärten Punkte wird die weitere Betrachtung des Ferienclubs und des Buchungsvorgangs in den nächsten Kapiteln auf die letzte der genannten Möglichkeiten zur Buchung eines Angebotes beschränkt.

4. Entwurf

4.1. Einleitung

In diesem Kapitel wird darauf eingegangen, wie die im vorderen Kapitel beschriebenen Probleme und Anforderungen in dem geschilderten Ferienclub gelöst und umgesetzt werden können. Es soll ein Entwurf entwickelt werden mit folgendem Geschäftsprozess:

Im ersten Schritt die Angebote vom Fremdanbieter abholen (polling oder on demand). Dem Gast die Angebote im Ferienclub zur Verfügung stellen. Im nächsten Schritt kann der Gast beim Hotel ein Angebot (z. B. Autovermietung) reservieren. Die damit verbundenen Formulare können von dem Gast offline auf dem Gerät vorbereitet werden bevor die Reservierung abgeschickt wird. Das Hotel übernimmt die Reservierung und die weitere Organisation (z. B. bei der Autovermietung, den Ort der Abholung oder die Übergabe der Wagenschlüssel). Das Hotel informiert den Gast über den Status (erfolgreich, oder nicht verfügbar). Weiterhin wird die mögliche Architektur des Ferienclubs vorgestellt.

Die Middleware seht im Zentrum des Entwurfes. Die gesamte Kommunikation soll darüber abgewickelt werden. Es wird dabei unterschieden zwischen der Middleware für die Kommunikation zwischen Personal Digital Assistent (PDA), Hotelserver und Hotel, Fremdanbieter.

Ziel des Kapitels ist es, einen Entwurf vorzustellen, der Vorlage für die Prototypen ist, mit denen Mono als Plattform für den Einsatz vom Middleware auf mobilen Endgeräten untersucht wird.

4.2. Architektur

Die hier erklärte Architektur soll als Basis für die weitere Betrachtung dienen und die Umgebung festlegen, die im Ferienclub genutzt wird (siehe Abbildung 4.1).

Es sind dabei die Strukturen des Hotels und der weiteren Anbieter vereinfacht worden. Das Hotelnetzwerk, ist ein Funknetzwerk, bestehend aus den PDAs und dem Hotelserver. Die Verbindung von Hotelserver zu den Anbietern (Autovermietung, Restaurant, Sportangebot) wird über das Internet realisiert. Dargestellt wird nur die äußere Kommunikationsschnittstelle

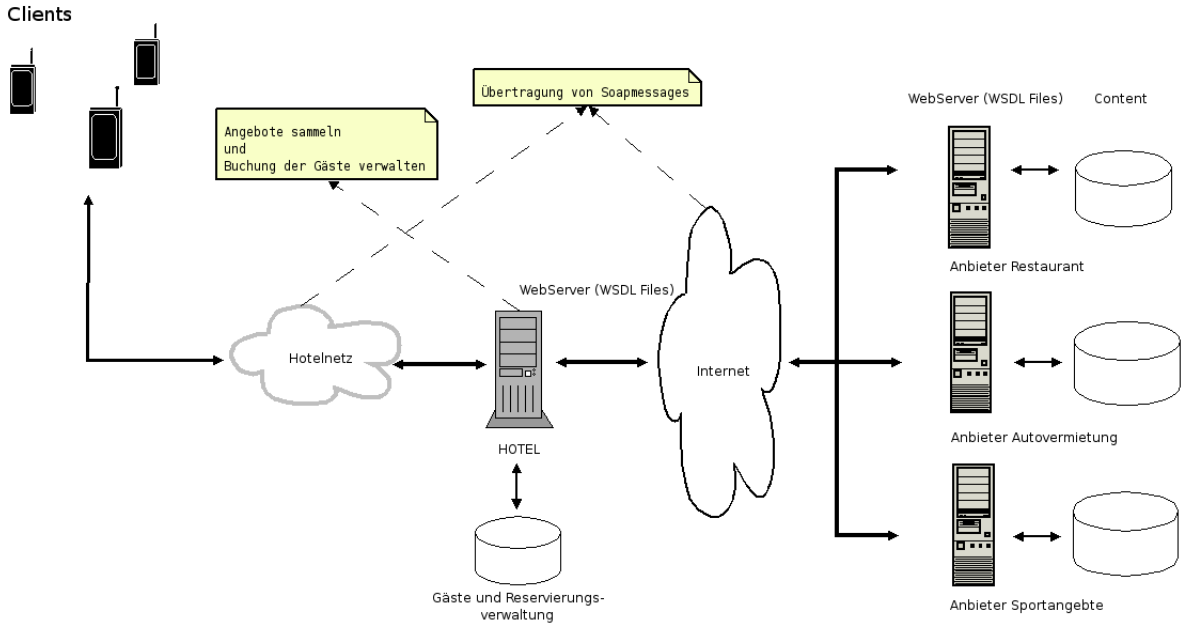

Abbildung 4.1.: Komponentendarstellung des Ferienclubs

und eine vereinfachte Version des Backends dieser Server. Vernachlässigt werden zudem die Sicherheitskomponenten, die in dem Netzwerk benötigt werden. Eine genauere Analyse ist nachzulesen in den Untersuchungen von Andre Lüpke “Entwurf einer Sicherheitsarchitektur für den Einsatz mobiler Endgeräte” [Lüpke 2004] und Lars Mählmann “Sichere Übertragung im WLan mit mobilen Endgeräten (speziell unter Linux)” [Mählmann 2004].

Die PDAs stellen den Gästen die Anwendungen zum Buchen bereit. Die Clients sind als *rich Client* ausgelegt. Sie sollen neben der Darstellung die Teile der Anwendungslogik bereitstellen. Die Anwendungslogik soll die Reservierungsangebote und die jeweilige Buchung auf dem Client verwalten, solange das Ferienclubnetzwerk außer Reichweite ist. Ein Beispiel könnte folgendermaßen aussehen: Der Gast ruft alle Angebote beim Ferienclub ab und möchte aber nur die Angebote für Mietwagen erhalten.
Dafür müssen nun auf dem Client die Daten gefiltert und die Darstellung aufbereitet werden. Der Gast bekommt nur den für ihn entscheidenden Teil der Angebote. Um die restlichen Angebote für den Gast zu einem späteren Zeitpunkt noch zur Verfügung stellen zu können, müssen auf dem Client die Daten vorgehalten werden.
Weiterhin muss eine Schnittstelle vorhanden sein, die bei Verbindung die Daten vom Ferienclubserver anfordert und abschickt. Auf Grund der oben geschilderten Anwendung ist es in diesem Zusammenhang nicht möglich, eine serverseitige Lösung zu benutzen, in dem der Hotelserver die komplette Anwendungslogik besitzt. Der Client dient nur zur Darstellung der Daten in einem Webfrontend (thin client Model). Ein weiterer Grund ist die Verfügbarkeit,

wenn der Gast sich ausserhalb des Clubs befindet, können die Angebote nicht eingesehen werden und sind nicht nutzbar. Die Anforderung aus dem Szenario, dass der Gast immer auf die Angebote zugreifen kann, wäre nicht erfüllt.
Es wird eine Schnittstelle benötigt, die auf mobilen Endgeräten verfügbar ist und diese muss in das bestehende Netzwerk integrierbar sein. Diese Schnittstelle wird auf Seiten des Ferienclubservers mit Hilfe von Web-Services angeboten. Die Clientanwendung ruft die Services auf und erhält die Angebote des Clubs oder kann mit Hilfe von Web-Services die Reservierungsdaten übergeben. Es können über diese Schnittstelle verschiedene mobile Clients bedient werden ohne die Anwendungslogik auf der Serverseite ändern zu müssen.

Die verschiedenen Anbieter bieten Leistungen über das Internet an und arbeiten zusätzlich mit dem Hotel zusammen. Die Server der einzelnen Anbieter können alle unterschiedliche Plattformen besitzen, wenn es um die interne Struktur und die Angebote geht. Entscheidend ist die standardisierte Kommunikation die für alle gleich sein muss, ebenso für den Ferienclub. Die Anbieter können alle unterschiedliche Geschäftsabläufe haben, um Reservierungen und Abrechnungen mit den Kunden abzuwickeln. Hinzu kommt die unterschiedliche Hard- und Software, die bei Ihnen eingesetzt wird. Es muss also eine Schnittstelle nach außen geben, die dieses berücksichtigt. Es muss möglich sein, die eigenen Daten den Kunden bereitzustellen. Zusätzlich dessen Antworten entgegen zu nehmen und innerhalb des eigenen Systems weiterverarbeiten zu können, unabhängig davon, woher die Daten kommen. Diese Schnittstelle sollte möglichst generisch sein, sodass z. B. nicht nur ein bestimmter Ferienclub darauf zugreifen kann, sondern alle Ferienclubs mit denen die Anbieter zusammenarbeiten. Es muss ein gemeinsames Protokoll geben, die allen Nutzern zur Verfügung steht und in einer heterogenen Landschaft benutzt werden kann. An dieser Stelle ist ein Einfluss auf die Software oder Plattform nicht möglich, da diese nicht zum Ferienclub gehört und nicht für diesen veränderbar ist.

Der Ferienclubserver ist die Schnittstelle zwischen Gast und Anbieter. Dieser dient zur Verwaltung der Gäste, der Planung, der Angebote des Hotels, Interna und der Abrechnung. Es ist der wichtigste Bestandteil des Netzwerkes. An dieser Stelle werden die Angebote der Anbieter eingeholt und aufbereitet, um diese dem Gast zur Verfügung zu stellen. Der Server stellt die Middleware über den die gesamte Kommunikation und Koordination abläuft. Es muss dabei eine Persistenz einbezogen werden, die einzelne Schritte von Reservierungen sichert und somit bei Problemen nachvollziehbar macht. Der Server muss eine Transparenz der Heterogenität der verschiedenen Systeme ermöglichen.

Das Hotel muss die beiden verschiedenen “Welten” verbinden. Der Server dient einmal als Server für die PDAs und muss eine dementsprechende Schnittstelle zur Verfügung stellen. Er benutzt gleichzeitig selber die Schnittstelle der Fremdanbieter als Client um dort Informationen abzurufen und Reservierungen zu buchen.

Grundlegende Architektur

Der Ferienclub lässt sich in vier Bereiche aufteilen (siehe 4.2):

1. Die Präsentationsschicht über die die Gäste die Angebote nutzen. Der Personal Digital Assistent stellt dem Gast die Angebote zur Verfügung und der Gast nutzt die Anwendung zum Buchen der verschiedenen Angebote. Der Client besitzt eine Schnittstelle zur Middleware für die Kommunikation mit den Hotelserver.

2. Die Webservices dienen als Technologie um die Clients mit Server, bzw. Ferienclub- und Anbieterserver zu verbinden. Für Transaktionen sollen die speziellen Web-Service Protokolle genutzt werden [Little und Freund 2003].

3. Die Middleware dient zur Kommunikation der oben geschilderten Komponenten. Die Kommunikation erfolgt über http bzw. https und SOAP Nachrichten.

Zur Umsetzung wird Mono als Plattform dienen und C# als Programmiersprache. Mit dem Einsatz von Mono erhält man die Möglichkeit zwischen verschiedenen Betriebssystemen (z. B. Linux, Windows) zu wählen und erhält eine gewisse Plattformunabhängigkeit. Mit C# gewinnt man die Vorteile einer objektorientierten Programmiersprache.

Mono muss auf den einzelnen Komponenten verschiedene Aufgaben erfüllen um in diesem System eingesetzt zu werden. Eine erste Aufgabe an Mono ist die Bereitstellung der Plattform auf verschiedenen Systemen. Anbieter und Ferienclub müssen sich auf Web-Services als gemeinsame Kommunikationstechnologie einigen. Es ist nicht zwingend erforderlich mit der selben Plattform zu agieren (z. B..NET und Java). Das Hotel und die Anbieter müssen gemeinsame Standards nutzen für die Sicherheit, die Transaktionen und die Informationen zur Reservierung. Der Ferienclub stellt intern einen Serverdienst bereit (über Web-Services), mit dem das mobile Endgerät auf die Dienstleistungen des Hotels zugreifen kann.

Der Client

Der Client besteht aus einer graphischen Benutzeroberfläche, die Angebote, Reservierungen und zusätzliche Informationen auf dem Client darstellt. Die Daten werden temporär gespeichert bis eine Verbindung zum Server besteht. Die Anwendungslogik baut die Verbindung zum Server auf, sorgt für die Darstellung der Daten auf dem PDA und verwaltet sie auf dem Handheld. Die Anwendung benutzt hierbei Web-Services zur Kommunikation über eine Funkverbindung.

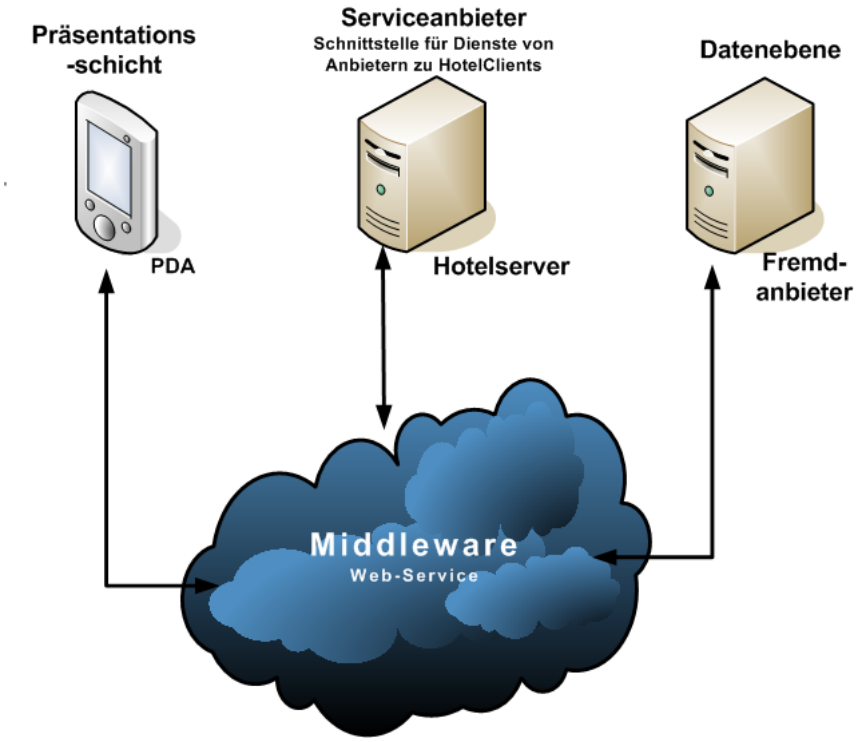

Abbildung 4.2.: Darstellung der verschiedenen Bereiche

Der Hotelserver

Der Hotelserver 4.3, der als Verteiler dienen soll, hat über die Web-Services eine Schnittstelle nach aussen. Er arbeitet als Provider für die mobilen Endgeräte. Über die Schnittstelle bietet der Ferienclub dem Gast die Möglichkeit Angebote abzufragen und dessen Reservierungen entgegen zu nehmen. Derjenige der die Dienste nutzt, sieht dabei nur diese Schnittstelle und benötigt kein Wissen darüber, was dahinter liegt. Intern besitzt der Server die vollständige Anwendungslogik, die nötig ist um die Verwaltung der Angebote und der Abrechnung zu übernehmen. Es sind dort noch weitere Module enthalten, wie die Verwaltung des Hotels, die als internal service bezeichnet werden.
An dieser Stelle werden die verschiedenen Angebote aufbereitet um sie den Gästen nach aussen über einen Web-Service zur Verfügung zu stellen. Die internal services sind noch weiter verzweigt und haben wiederum verschiedene Schichten, wo es ein Backend geben kann, das ein Replikat der Angebote der Anbieter vorhält, um Anfragen auf Zweitangebote schnell zu gewährleisten. Dort werden ebenfalls die Reservierung der Gäste mit dem Status der Buchungen persistent festgehalten.

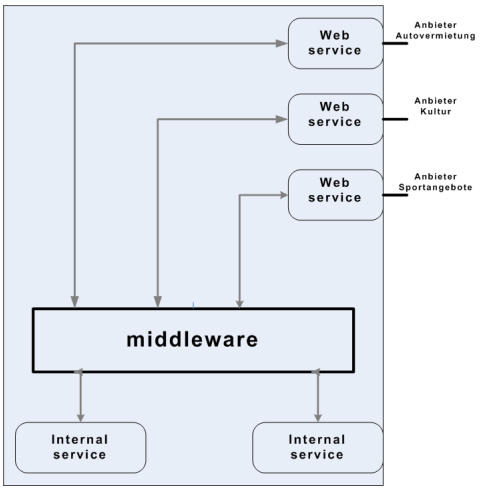

Abbildung 4.3.: Der Hotelserver als Provider (Seite 135,[Gustavo Alonso und Machiraju 2004])

Wenn der Hotelserver mit den Anbietern kommuniziert ändert sich die Darstellung 4.4: Das Hotel wird zum Client, der die Web-Services der verschiedenen Anbieter nutzt, um Angebote abzufragen oder Reservierungen im Auftrag der Gäste vorzunehmen. Die Angebote werden durch die Web-Services der Anbieter “abgeholt” und danach in internal services übersetzt.

Die Kommunikation im Hotelnetzwerk findet über ein Funknetz und mit den Anbietern über das Internet statt.

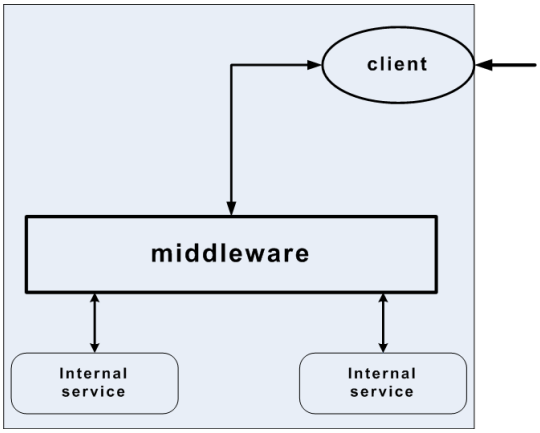

Abbildung 4.4.: Der Hotelserver als Client (Seite 135,[Gustavo Alonso und Machiraju 2004])

Die Anbieter

Die Anbieter sind dem Hotelserver (4.3) ähnlich. Sie sind zu dem Hotel und weiteren Anbietern und Nutzern inhomogen und stellen über die Web-Services ihr Angebot bereit. Anders als der Hotelserver arbeiten sie nur in der Serverrolle und nicht zusätzlich als Client. Die Web-Services bieten die Benutzung der Anwendungslogik über das Internet und kapseln das Interne nach aussen ab. Die Anwendung ist nutzbar über die Web-Services aber Wissen über die Struktur und die Anwendung selber sind nicht nötigt. Damit ist es möglich viele verschiedene Kunden zu erreichen ohne für jede einzelne spezielle Änderung vorzunehmen.

4.3. Design

Ablauf der Buchung

In der unten gezeigten Darstellung 4.5 wird gezeigt, wie die Kommunikation zwischen den Gästen, Hotel und weiteren Anbietern aussehen wird.

Es dienen die PDAs als Clients, die Angebote anfordern und als Endanwender nutzen. Die Komponente, die die Anfragen beantwortet, dient dabei als Server (in diesem Fall der Hotelserver) für die PDAs und gleichzeitig als Client, wenn die Angebote von Fremdanbietern angefordert werden, die an die Clients, im Hotelnetzwerk, weitergegeben werden. Die Fremdanbieter, wie die dargestellte Autovermietung, sind Server die Anfragen beantworten und bearbeiten.

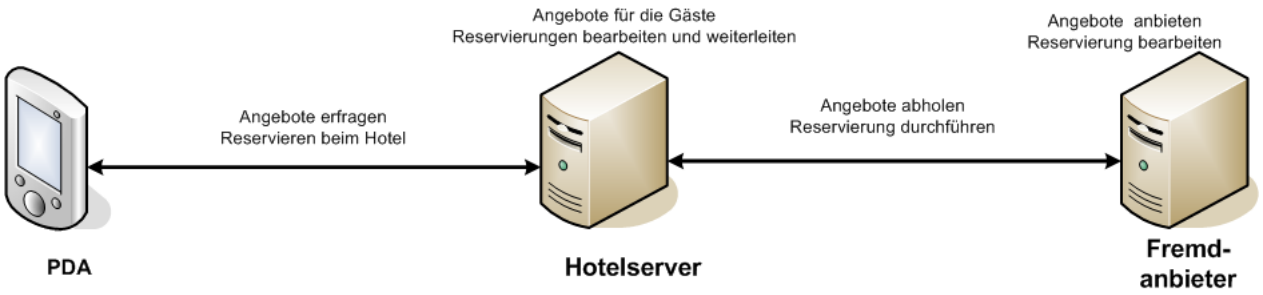

Abbildung 4.5.: Grobe Darstellung der Komponenten

Beispielhaft ist hier der Ablauf der Buchung bei einem Fremdanbieter über das Hotel skizziert (4.6).

1. Der Gast erstellt eine Anfrage.
2. Die Verbindung zum Netzwerk wird aufgebaut.
3. Es wird dem Service die Information gegeben, dass es eine Buchungsanfrage für eine angebotene Dienstleistung möchte.
4. Der Dienst beim Hotel sucht den entsprechenden Anbieter und gibt die Anfrage weiter.
5. Die Anfrage wird dort auf Verfügbarkeit und Gültigkeit überprüft.
6. Der Anbieter sendet die Antwort auf die Anfrage zurück.
7. Das Hotel lässt sich bei einer positiven Rückmeldung des Anbieters diese Anfrage vom Gast noch einmal bestätigen.
8. Der Gast bestätigt oder lehnt ab. Bei einer Ablehnung wird der Buchungsvorgang beendet.
9. Bei Zusage sendet das Hotel die Reservierung an die Fremdfirma. [1]
10. Dort wird diese bearbeitet und eine Bestätigung an das Hotel gesandt.
11. Erreicht diese Antwort das Hotel wird die Buchung rechtsgültig.
12. Das Hotel schickt eine Bestätigung der Reservierung an den Gast. Zusätzlich hat der Gast die Möglichkeit sich beim Hotel direkt über den Stand der Reservierung zu informieren. Denkbar wäre dabei die Bearbeitung über einen personalisierten Terminplan, der alle Informationen über den jeweiligen Gast vorhält. Marco Presch hat in seiner Diplomarbeit solch einen Kalender entworfen und untersucht. (siehe [Bresch 2004]).

[1] An dieser Stelle ist ein Abbruch der Transaktion durch den Gast nicht mehr möglich und ein Verbindungsabbruch kann nur noch durch die Server (Hotel, Autovermietung) oder durch das Netzwerk (eher unwahrscheinlich, wenn das Netz richtig dimensioniert wurde) geschehen.

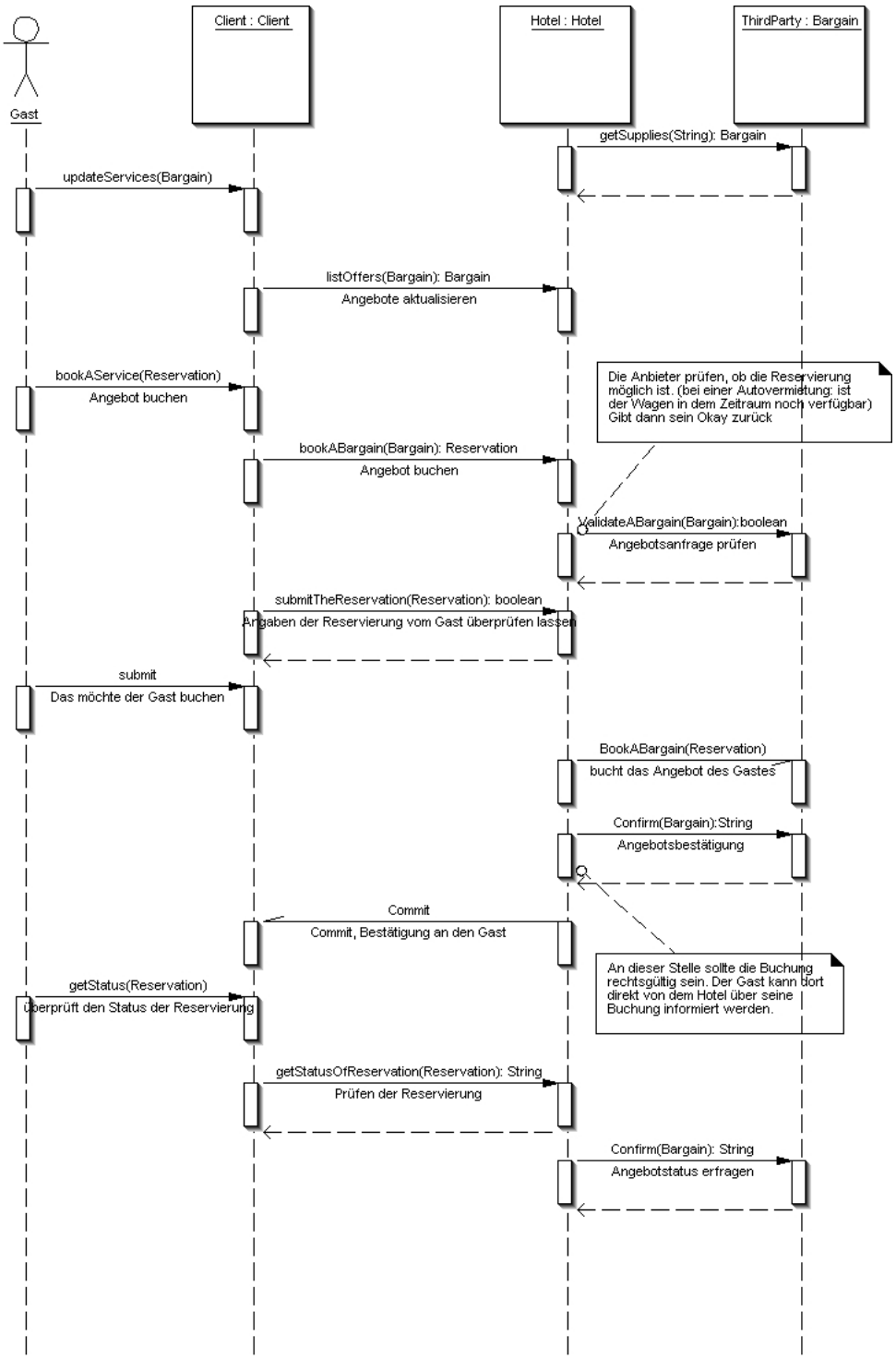

Abbildung 4.6.: Ablauf der Buchung vom Gast bis zum Anbieter

Software Entwurf der Web-Services

Im Folgenden sind die einzelnen Klassen aus dem Sequenzdiagramm 4.6 aufgegriffen und als Klassendiagramm verdeutlicht. Es zeigt die Web-Services von den einzelnen Komponenten und die Abhängigkeit untereinander. An Hand der vorhergegangen Architektur 4.5 sind nun die einzelnen Bestandteile in Klassen dargestellt. Es sind nur die Methoden enthalten, die benutzt werden sollen um Angebote zu lesen oder Reservierungen durchzuführen. Die Methoden aus der Client-Klasse rufen die Web-Services vom Hotelserver auf und beinhalten selber keine Services.

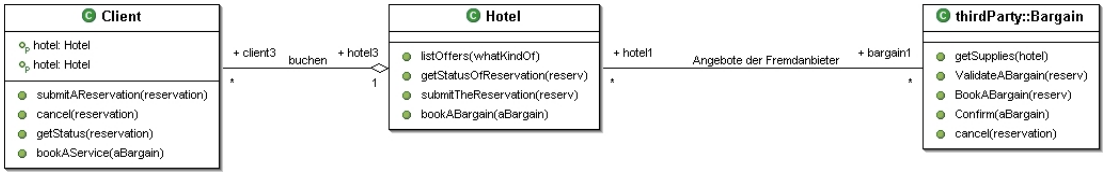

Abbildung 4.7.: Klassendiagramm der WS-Klassen und der Methoden

Die Methoden der Client-Klasse rufen die Web-Services auf mit denen die Anwendung die Angebote abrufen bzw. buchen kann. Das Hotel benutzt internal services, um die aktuellen Reservierungen der Gäste zu verwalten. Dort enthalten sind die Reservierungsinformationen der Anbieter und die dazugehörigen Daten der Gäste. Die Darstellung dieser Anwendung ist hier nicht enthalten. Die Klasse *Third-Party* repräsentiert die verschiedenen Anbieter. Die Anbieter stellen die Web-Services, mit denen ein Ferienclub die Angebote der Firmen abfragt und für Gäste Reservierungen vornimmt. Die Klasse *Third-Party* ist eine abstrakte Repräsentation der Anbieter, die verschiedenen Angebote zur Verfügung stellen.

Die einzelnen Web-Services

Die in dem Sequenz- 4.6 und Klassendiagramm 4.7 gezeigten Methoden der Web-Services sollen genauer beschrieben werden. Es soll den Ablauf zwischen den einzelnen Komponenten verdeutlichen und die Schwierigkeiten und Probleme aufzeigen, die in diesem Zusammenhang entstehen. Die folgenden Methoden werden benötigt um den Gästen das Buchen über einen PDA anzubieten.

Web-Services des Anbieters

Die Web-Services, die von den Anbietern dem Hotel zur Verfügung gestellt werden um Angebotsinformationen abzurufen und Reservierungen zu buchen.

- `getSupplies(String hotel)` mit diesem Aufruf kann ein Kunde, hier das Hotel, die Angebote abholen und aktualisieren. Das Hotel gibt bei der Anfrage den Namen seines Hotels mit an um dem Anbieter zu übermitteln, welche Angebote er haben will. Die einzelnen Angebote können für die einzelnen Kunden unterschiedlich sein, je nach Anforderungen und Vertragsverhandlungen. Die verschiedenen Angebote werden vom Anbieter als Objekt (Bargain) übermittelt. In diesem Objekt sind dann enthalten in welchen Zeitraum dieses Angebot gültig ist, die Kosten und als Beispiel bei der Autovermietung spezielle Informationen, wie die verschieden Autotypen.

  Der Kunde kann beliebig oft diese Angebote aufrufen. Er erhält immer die aktuellen Informationen, die von den internen Komponenten des Anbieters zur Verfügung gestellt werden.

- Hat der Gast einen Reservierungswunsch für ein spezielles Angebot, prüft das Hotel mit Hilfe von `(ValidateABargain(Bargain bargain))`, ob dieses Angebot gültig ist und zusätzlich noch verfügbar. Die Angebote sind begrenzt und wenn keine Plätze oder bei der schon angesprochenen Autovermietung keine Autos mehr erhältlich sind kann der Reservierung nicht entsprochen werden. Der Anbieter verlangt beim Aufruf dieses Services ein *Bargain Objekt*, was alle Daten enthält, welche für die Buchung nötig sind. Als einfache Referenz dient dabei das Objekt des Angebotes. Es müssen Informationen enthalten sein, wie z. B. das Datum, die Teilnehmerzahl, Kinder und weitere Daten; bei dem Beispiel der Autovermietung die Autoklasse oder Informationen über den Fahrer. Der Kunde bekommt als Rückgabewert eine einfache Zustimmung oder Ablehnung der Reservierungsanfrage. [2]

- Mit `BookABargain(Bargain aReservation)` wird die Reservierung vom Kunden "abgeschickt". Es wird das in der Methode `ValidateABargain(Bargain bargain)` geprüfte Angebot nun als Reservierung an den Anbieter übermittelt. Zusätzlich zu den Informationen welcher Kunde (z. B. Hotel) bucht, wird übermittelt wer die Rechnung und die Reservierungsbestätigung erhält. Zu diesem Zeitpunkt werden erstmals die personenbezogenen Daten des Gastes weitergegeben (z. B. bei der Autovermietung: Name, Führerscheininformationen und Geburtstag) [3].

- An Hand von `Confirm(String aReservationNumber)` kann eingesehen werden, welchen Status die Reservierung hat oder später die Daten der Reservierung noch einmal überprüft werden. Der Rückgabewert des Anbieters kann dann `aBargain` sein.

[2] Es wäre vielleicht möglich bei Ablehnung der Reservierung Alternativen dazu anzubieten, wie anderer Wagentyp ... das könnte über " Agenten" laufen.

[3] Diese Transaktion könnte über ein *CompletionWithAck-Protokoll* durchgeführt werden. Man benutzt dieses Protokoll um eine Applikation, hier das Hotel, über den Ausgang der Transaktion zu informieren. Nach Abschluss der Transaktion informiert der Anbieter das Hotel über den Status (Zu- oder Absage) Seite 201,[Knuth 2003] und Seite 228,[Gustavo Alonso und Machiraju 2004]

- Durch den Web-Service `Cancel(String aReservationNumber)` kann der Kunde eine Buchung stornieren. Dieser Service ermöglicht nicht die automatische Stornierung eines Angebotes sondern gibt nur den Wunsch der Stornierung wieder. Der Anbieter prüft beim Eingang der Stornierung, inwieweit und mit was für einem Kostenaufwand der Stornierung entsprochen werden kann. Der Anbieter sendet dann eine Antwort nach Prüfung der Anfrage, die dann auch direkt vom Personal kommen kann, da die Services solche Probleme nur bedingt prüfen können.

Die Vorgabe der zu benutzenden Methoden zur Buchung beim einzelnen Anbieter ermöglicht die einfache Verarbeitung der Anfrage, solange die gemeinsamen festgelegten Schnittstellen vollständig implementiert wurden. Wenn der Anbieter intern etwas ändert muss dies nicht zusätzlich allen Kunden mitgeteilt werden, da die Schnittstelle (das Interface) sich nicht ändert. Im schlimmsten Fall ändern sich die Web-Service, wodurch die Kommunikationsschnittstelle nicht mehr kompatibel zu der Vereinbarung wäre.

Methoden für das Hotel

Damit die Gäste mit dem PDA buchen können, muss das Hotel als Server einige Web-Services anbieten. Die hier aufgezeigten Web-Services erfüllen dabei die verschiedenen Aufgaben. So kann der Gast Angebote einsehen, sowie reservieren.

- `ListOffers()` holt alle angebotenen Angebote vom Ferienclub auf den PDA und aktualisiert gegebenenfalls die dort schon angelegten Angebote. Die Aktualität der Informationen, die der Gast erhält, sind abhängig von den Aktualisierungen des Ferienclubs bei den Fremdanbietern.
- `BookABargain(Bargain reservation)` wird vom Hotel benutzt um Reservierungen des Gastes anzunehmen und intern weiterzuverarbeiten. Im nächsten Schritt wird beim Anbieter eine Anfrage auf Verfügbarkeit gestellt.
  Das Hotel speichert die Buchung des Gastes mit dessen Informationen in der eigenen Datenbank zwischen, um dem Gast ngebote bei weiteren Anfragen oder Stornierung persistent vorhalten zu können bei weiteren Anfragen oder Stornierung. Es ist dem Ferienclub darüber möglich die einzelnen Schritte bei der Buchung nachvollziehbar zu halten um bei Problemen darauf zurückgreifen zu können. Für den Gast läuft die Buchung völlig transparent ab. Dieser erfährt nichts über die einzelnen Zwischenschritte auf dem Hotelserver.
- Nach der Prüfung auf Verfügbarkeit erhält der Gast durch das Hotel eine Benachrichtigung über die Anfrage und muss seine Reservierung mit `SubmitTheReservation(Reservation reservation)` bestätigen. Es

wird ein einfacher Booleanwert übermittelt, der noch einmal bestätigt, dass alle Informationen korrekt sind und man mit der Buchung des Angebotes einverstanden ist. Nach Eingang der Zustimmung bucht das Hotel das Angebot verbindlich beim Anbieter. Der Gast bekommt dann die Reservierungsbestätigung des Anbieters vom Hotel über einen `Commit` weitergereicht. Das Hotel berechnet die Kosten für das Angebot und addiert diese auf die bestehende Zimmerrechnung hinzu. Am Ende des Aufenthaltes wird dann die Gesamtabrechnung vorgenommen.

- Mit `GetStatusOfReservation(String reservationNumber)` erhält der Gast die Information über die von ihm gewünschte Reservierung. Das beinhaltet den Status falls er noch keine Antwort über die Buchungsanfrage erhalten hat. Eine weitere Information wäre z. B. der Preis für das gebuchte Angebot oder, bei einer Autovermietung, der Autotyp. Das Hotel übergibt die Daten mit Hilfe eines *Reservierungsobjektes* in dem alle Information zusammengefasst sind und von der Applikation auf dem PDA aufbereitet werden um sie dem Gast darzustellen.
- Falls ein Gast eine Buchung stornieren möchte kann er die Anfrage an das Hotel mit `Cancel(String reservationNumber)` stellen. Das Hotel nimmt die Anfrage entgegen, prüft diese auf Korrektheit und leitet sie an den Anbieter weiter. Die Bestätigung zur Stornierung und die entstehenden Kosten teilt das Hotel dem Gast nach Eingang der Antwort umgehend mit. Die Information könnte auf Grund der Stornierung persönlich erfolgen, da der Anbieter diese Fälle nicht automatisiert bearbeitet, sondern eher durch seine Sachbearbeiter, die dann direkt mit dem Hotel kommunizieren.

Methoden auf dem mobilen Endgerät

Die verschiedenen Methoden auf dem Client dienen dazu die Web-Services auf dem PDA in die Anwendung einzubinden. Der Gast benutzt eine grafische Oberfläche um die Angebote “abzuholen” und gegebenfalls zu buchen.
Die Methode `GetBargains()` liefert die Angebote und `bookABargain(Bargain aBargain)` bucht die Reservierung, die mit `SubmitAReservation(Reservation reservation)` bestätigt wird.
Zu einem späteren Zeitpunkt kann mit `GetStatusString reservationNumber` eine Reservierung abgefragt oder mit `Cancel(String reservationNumber)` die Reservierung storniert werden.

Bei diesem Ablauf treten mehrere Schwierigkeiten auf, die eine Erweiterung des Klassendiagramms nötig machen.
Diese ergeben sich aus der technischen Basis von Web-Services und der fachlichen Implementierung der Anwendungslogik. Bei der Benutzung von Web-Services ist zu beachten, dass diese zustandslos sind. Soap erlaubt auf Grund des Designs keine Übermittlung von

Objekt-Referenzen. Dadurch, dass Web-Services zustandslos sind, kann der Client eines Web-Services nicht erwarten, dass ein vorher angefragt und verwendetetes Objekt auf Seiten eines Webserviceanbieters bei einem zweiten Aufruf erreichbar ist.
Bei der fachlichen Implementierung soll vermieden werden, dass interne Abläufe eines Geschäftsprozesses von aussen einsehbar sind. Die Anwendungslogik soll vor dem Benutzer des Web-Services gekapselt werden. Dieses trifft auf das Hotel zu, wie auch auf die Anbieter.
Ein Beispiel ist der Preis für das Angebot einer Autovermietung. Der Preis setzt sich aus Wagentyp und Zeitraum zusammen. Das übertragene Objekt enthält die Daten Wagentyp und Zeitraum, kann aber nicht die Preisberechnung vornehmen, weil ein solcher Funktionsaufruf über Web-Services nicht verfügbar ist. Der Funktionsaufruf ist über die Web-Services nicht möglich und weiterhin will der Anbieter nicht die Information “preisgeben”, wie sich die Mietgebühr für ein Auto berechnet.

Im nächsten Kapitel soll die Möglichkeit gezeigt werden, wie diese Schwierigkeiten umgangen bzw. gelöst werden kann.

Redesign

In diesem Abschnitt soll eine Lösung für die im vorigen Kapitel beschriebenen Probleme dargelegt und erläutert werden.

Soll nun eine Reservierung getätigt werden, geht diese Anfrage an den dafür vorgesehenen Anbieter, hier z. B. eine Autovermietung 4.8.

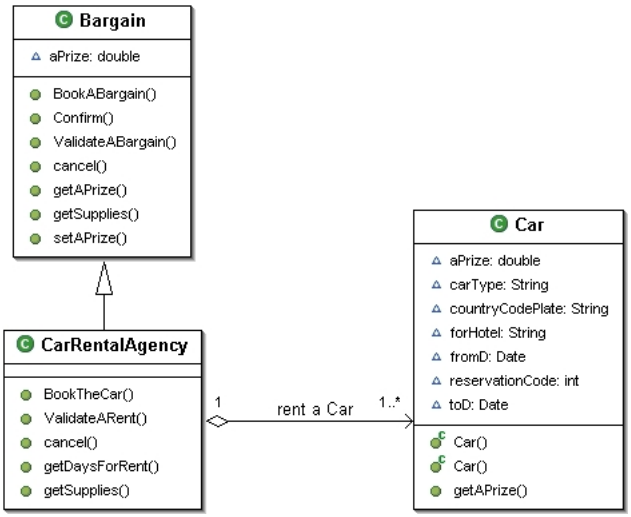

Abbildung 4.8.: Klassendiagramm zum Buchen eines Autos

Die Daten über ein Auto werden zusammengetragen in der Klasse *Car* und über die Klasse *CarRentalAgency* zu einem Angebote zusammengestellt. Dort wird zusätzlich der Preis berechnet, der sich aus verschiedenen Faktoren zusammensetzt (Wagentyp, Dauer des Verleihes und Kundendaten). Dieses Angebot wird an den Kunden weitergegeben und nach einer Überprüfung durch den Kunden gebucht.

Um das Angebot dem Kunden (z. B. Ferienclub) durch die Web-Services übergeben zu können, dürfen aber nur “einfache” Daten übergeben werden. Es müssen die abgeleiteten Werte in einen einfachen Datentyp übernommen werden. Die Realisierung könnte wie folgt Aussehen 4.9:

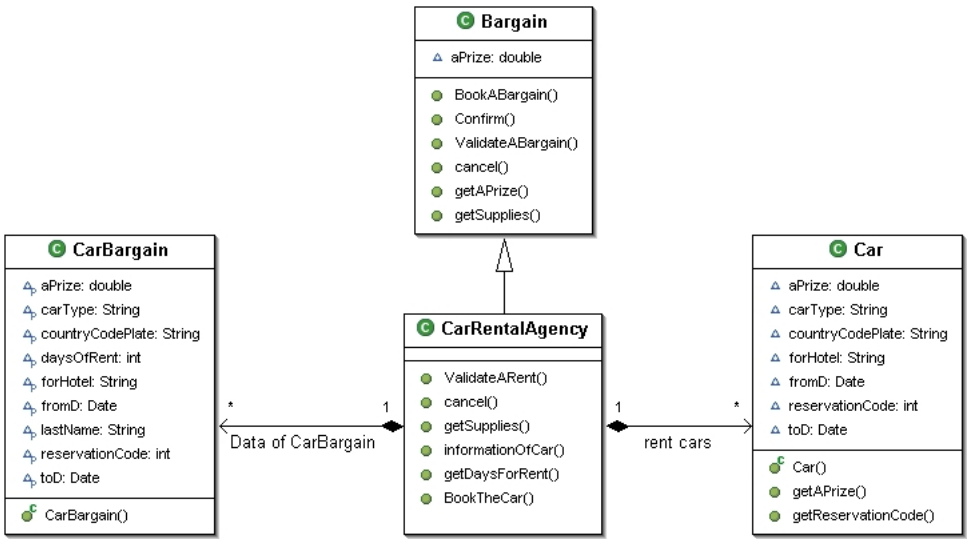

Abbildung 4.9.: Klassendiagramm zum Buchen eines Autos mit zusätzlichem “Datenobjekt”

Es gibt weiterhin die Klasse *Car*. Die Berechnung der zusätzlichen Informationen wie Preis und Verleihdauer werden in der Klasse *CarRentalAgency* berechnet. Die gesammelten Information werden an das Datenobjekt *CarBargain* übergeben, was aus einfachen Datentypen besteht. Dieses Objekt wird benutzt um die Angebote bei `BookABargain` an das Hotel zu übergeben.

Da das Hotel die einzelnen Reservierungsanfragen benötigt um die Abrechnung mit dem Gast vorzunehmen und den Status der einzelnen Reservierungen zu kennen, ist es nötig die Informationen “zwischen zu speichern”. Somit ist das Hotel in der Lage, bei Problemen mit den Verbindungen zum Gast oder zu den Anbietern die Daten jederzeit ein weiteres Mal zu versenden oder bei Nachfragen bereitzustellen. Eine einfache Lösung wäre, eine Klasse anzulegen, wo ein Objekt die Daten des Angebotes für einen Gast sowie die Daten des Gastes im Hotel besitzt (Abbildung 4.10). Über eine Reservierungsnummer, die eindeutig ist, kann das Objekt jederzeit für weitere Nachfragen oder Veränderungen aufgerufen und verändert

werden. Es enthält dabei nur die Daten des Gastes, wie Name oder Zimmernummer. Die Daten der Anbieter bestehen aus dem Datenobjekt *Bargain* welches von den verschiedenen Fremdanbietern wie oben beschrieben übergeben wird. Ist nun ein Angebot abgearbeitet und die Transaktion zwischen Hotel und Anbieter erfolgreich abgeschlossen, kann es z. B. persistent in einer Datenbank abgespeichert werden und später für die Abrechnung mit dem Gast wieder aufgerufen werden.

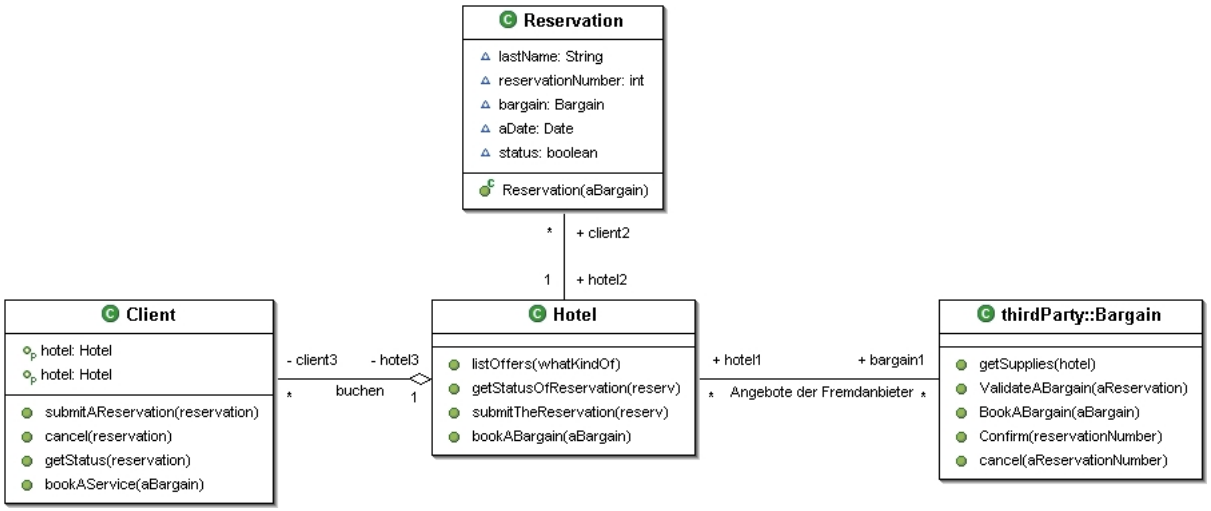

Abbildung 4.10.: Klassendiagramm zum Vorhalten der Reservierungsinformationen beim Ferienclub

Die einzelnen Variablen bestehen aus:

- den Daten des Gastes (`lastName, reservationNumber`)
- den Informationen zu der Buchung des Gastes (`bargain, aDate`)
- sowie dem jeweiligen Status (`status`)

diese Werte sind nur ein Beispiel und mit weiteren Daten zu erweitern, wobei es auf die jeweiligen Angebote und Gäste ankommt. Die Informationen sind von einfachen Datentypen und über Web-Services nutzbar.

4.4. Design-Pattern

Die bisherige Ausführung beschäftigte sich hauptsächlich mit der Umsetzung der Architektur und der Umsetzung des Sequenzdiagramms 4.6 aus dem vorigen Kapitel. Um aber Anforderung wie Wartbarkeit, Erweiterbarkeit und Skalierbarkeit nachzukommen, sollten die bisherigen Entwürfe nachgebessert und erweitert werden.

Die Architektur des Systems lässt sich nicht eindeutig in verschiedene Teilbereiche gegeneinander abgrenzen, wie bei einer three-tier Architektur. Die Bereiche überlappen sich an einigen Bereichen, z. B. bei dem Client, wo zusätzlich zur Präsentationsschicht auch ein Teil der Anwendungslogik mit enthalten ist. Das Backend besteht aus mehreren verschiedenen Anbietern die alle dem Ferienclub (Anwendungslogik, Middleware) Daten zur Verfügung stellen und mit diesem kommunizieren. Um für diese Anwendung möglichst “einfache und schlanke” Schnittstellen zu erhalten kann auf verschiedene Design-Patterns, wie sie bei [Gamma Erich und Vlissides 1995] beschrieben sind zurückgegriffen werden. Eine Möglichkeit ist das Facade-Pattern um die Schnittstelle zwischen Ferienclub und Anbietern zu vereinfachen. Eine weitere Möglichkeit kann das Proxy-Pattern darstellen, welches an verschiedenen Punkten eingesetzt werden kann. Eine genauere Betrachtung der aufgeführten Punkte soll in dem nächsten Abschnitt folgen.

Design-Patterns auf Serverseite

Auf Seiten des Ferienclubs sind verschiedene Erweiterungen möglich um die Skalierbarkeit zu verbessern. Die Netzwerkauslastung sollte möglichst gering gehalten werden. Gleiches gilt für den Speicherbedarf des Clients bei der Abfrage von den verschiedenen Anbietern und Reservierungen. Die Netzwerkauslastung ist ein wichtiger Aspekt, wenn es darum geht den Gästen über Funknetze die Angebote anzubieten. Es sollten zusätzlich möglichst einfache Schnittstellen zur Verfügung gestellt werden um die Erweiterbarkeit zu fördern.

Proxy-Pattern

Ein Ferienclub kann mit beliebig vielen Anbietern zusammenarbeiten um dem Gast ein möglich breites Angebot von Freizeitaktivitäten und Ähnlichem anzubieten.
In der Grafik 4.7 werden die Anbieter abstrakt dargestellt mit der Klasse *ThirdParty*. Die Kommunikation des Ferienclubs (Servicenutzer) und der Fremdanbietern (Serviceanbietern) über ein gemeinsames Protokoll (z. B. http) und eine gemeinsame Schnittstelle (*ThirdParty*), bei der im Hintergrund des Serviceanbieters verschiedene Dienstleistungen benutzt werden, ist ein Beispiel für ein Proxy-Design-Pattern [Gamma Erich und Vlissides 1995]4.11.

Durch das Einsetzen eines Platzhalters (Proxy) auf der Servicenutzerseite werden die Einzelheiten (z. B. Protokoll) der Anbieterseite vor den Nutzern gekapselt. Der Proxy versucht möglichst alle Anfragen direkt abzuhandeln und nur die Nachrichten an den echten Serviceanbieter weiterzugeben, wenn er die Anfrage nicht selber beantworten kann. Durch dieses restriktive Verhalten des Proxys kann man die Performance (Speicherauslastung, Antwortzeiten) des Systems optimieren, da der “echte” Service erst aufgerufen wird, wenn dieser im verteilten System benötigt wird ([Sherman R. Alpert und Woolf 1998]).

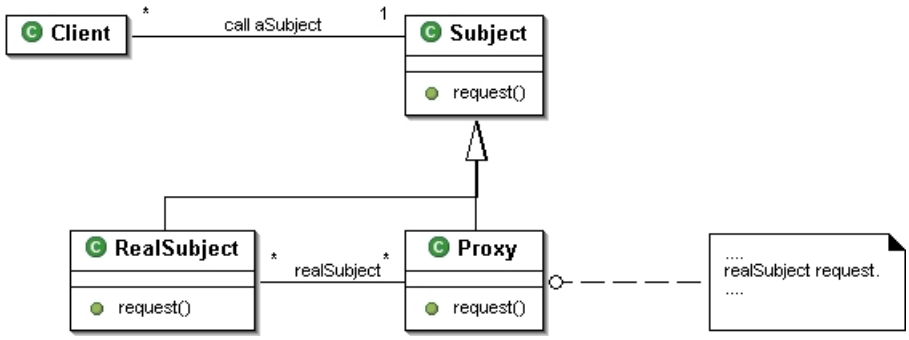

Abbildung 4.11.: Das Proxy-Pattern nach [Sherman R. Alpert und Woolf 1998]

Eine Umsetzung, wie sie in [Knuth 2003] beschrieben wird, ist hier in 4.12 abgebildet, typischerweise sind Web-Services Architekturen derart konstruiert.

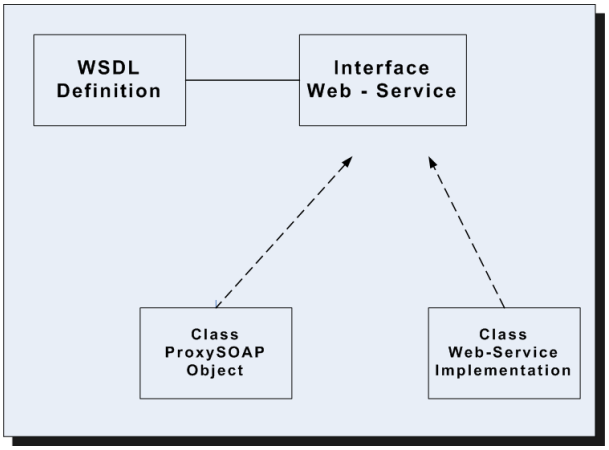

Abbildung 4.12.: Das Proxy-Pattern beim Einsatz mit Webservices

Die Umsetzung könnte dann folgendermaßen aussehen:

- Die Beschreibung der WSDL-Definition wird in einem *C# Interface* definiert.
- Das Interface wird von den Klassen ProxySOAPObject (Client) und des Web-Services (Server) implementiert.
- Diese beiden Klassen sind über das SOAP-Protokoll verbunden. Wird nun eine Methode aus dem Interface aufgerufen erfolgt ein SOAP Request, der die Verarbeitung durch die Web-Services aufruft und die Antworten zurückgibt.

Facade-Pattern

Ein Facade-Pattern definiert eine Schnittstelle, welche mehrere verschiedene Interfaces zusammenfasst. Durch die Zusammenfassung zu einer Schnittstelle wird die Benutzung vereinfacht. Mit Hilfe dieses Entwurfsmusters ist es möglich die unterschiedlichen Anbieter in einer Schnittstelle zusammenzufassen und auf der Ferienclubseite auf eine Schnittstelle zu reduzieren. Die Verarbeitungslogik und Geschäftsprozesse eines Serviceanbieters werden dahinter versteckt (z. B. PDA). Bei Änderungen der eigenen Umgebung muss diese Schnittstelle erhalten bleiben, bzw. sie muss mit eingebunden werden. An Hand einer Grafik 4.13 wird verdeutlicht, wie diese Fassade aufgebaut ist und benutzt werden kann.

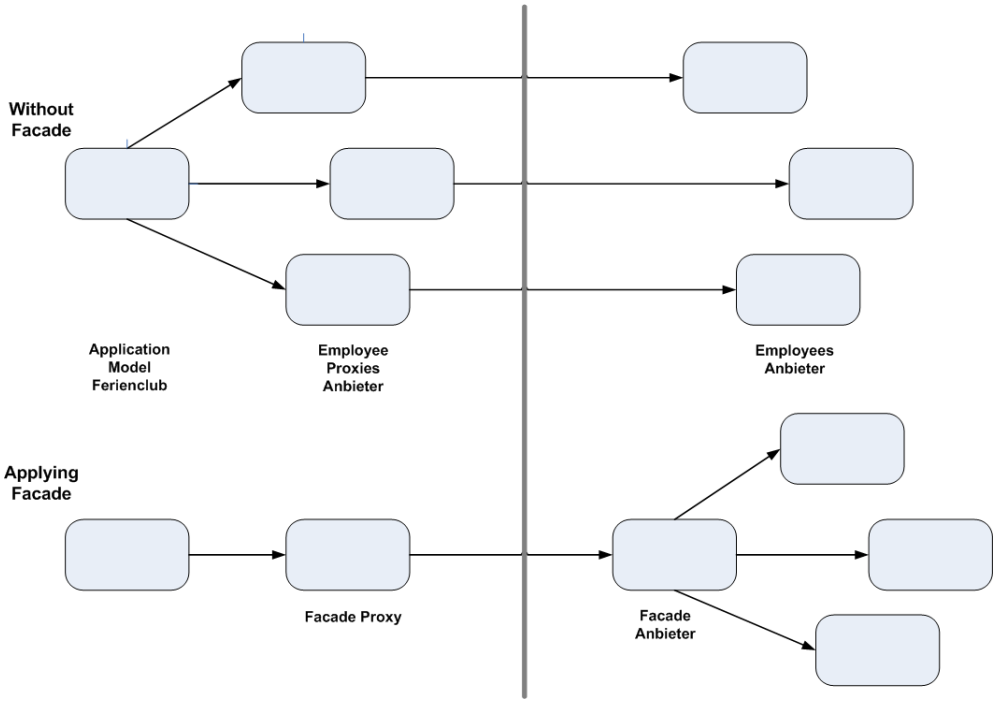

Abbildung 4.13.: Das Facade-Pattern [Sherman R. Alpert und Woolf 1998]

Im oberen Teil der Abbildung 4.13 ist der Ferienclub ohne das Pattern dargestellt. Es werden direkt die einzelnen Dienste der Anbieter aufgerufen. Die Netzwerkbelastung ist hoch und es gibt viele verschiedene Schnittstellen, die zu einer insgesamt großen, komplizierten und schwer zu verstehenden Schnittstelle führen. Mit dem Facade-Pattern wird ein Schnittstelle vorgegeben hinter der die einzelnen Aufrufe gekapselt werden. An diesem Punkt werden die Angebote über die jeweiligen Services aufgerufen (über den Proxy) und dem Ferienclub übermittelt. Zur Vereinfachung der Schnittstelle wird der Speicherplatz auf der Clientseite verringert und die Netzwerklast ebenfalls.

4.5. Betrachtung des Datenaustausches

Bisher wurde die Umsetzung der Web-Service beschrieben, wie sie im Sequenz- und den Klassendiagramm vorkommen. Es war dabei die Rede vom “Abholen der Angebote”. Aber wie soll das genau passieren? Weiterhin wurde beschrieben, dass man vom offline Modus in den online Modus wechseln kann und die Daten dann abholt oder Reservierungswünsche abschickt. Wie ist dieses möglich und was bietet Mono dafür an? An dieser Stelle soll kurz beschrieben werden, welche zusätzlichen Überlegung angestellt werden müssen und wie die Umsetzung aussehen könnte unter Mono.

Synchronisation der Daten

Der Abgleich der Daten wurde im oberen Kapitel (bei dem Ablauf), einfach als abholen bei den jeweiligen Servern bezeichnet. Aber in Wirklichkeit ist dieser Vorgang komplizierter und benötigt einige Überlegungen um eine Synchronisation der aktuellen Angebote zu gewährleisten.
Es muss entschieden werden, wer die aktuellen Angebote vorliegen hat, wann ein Angebot aktueller ist als das Bekannte beim Gast oder dem Hotelserver. Wie sollen dann die Daten abgeglichen werden und wie oft wird der Abgleich vorgenommen.

Bei der Synchronisation könnte folgendes Problem auftreten: Der Gast kann ein Angebot reserviert haben und erhält beim Abgleich die Information, dass dieses Angebot nicht mehr existiert. Solche Widersprüche sollten vermieden werden. Das Interesse des Gastes an bestimmten Angeboten muss berücksichtigt werden.

Bei dem Abgleich der Daten zwischen Hotel und Anbieter kann festgelegt werden, dass die aktuellsten Angebote bei den Anbietern liegen. Der Abgleich für diese Angebote könnte einmal am Tag passieren (z. B. in der Nacht). Das Hotel erreicht ein Grundgerüst an Angeboten, die es dem Gast vorstellen kann. Wenn ein Gast ein erweitertes Interesse an diesem Angebot hat, werden die aktuellen Angebote direkt abgeholt und die bisherigen überschrieben. Die Angebote werden dabei in einem einfachen Textformat übertragen und können dann einfach auf der gegenüberliegenden Seite weiterverarbeitet werden. Auf Grund der Menge an Daten und der vielfachen Anfragen durch die Gäste wäre eine Speicherung in einer Datenbank denkbar . Auf dem PDA könnten die Angebote in einer XML Datei hinterlegt werden, wo die gebuchten Angebote durch ein “tag” speziell gekennzeichnet werden können, ebenso wie die aktuellen Anfragen.
Für die hier geschilderte Testumgebung wird es reichen, Mono als Plattform für Web-Services zu untersuchen. Wenn allerdings die Realität betrachtet wird, hat die Problematik der Synchronisation eine gesteigerte Bedeutung. Die vermehrte Verwendung von mobilen

Endgeräten führt zu weiteren Problemstellungen, wie z. B. Komplexität durch Kalenderfunktion und die damit verbundene Terminplanung.

Verbindungen beim Endgerät

Die Beschreibung des Wechsels zwischen dem off- und online Modus wurde bisher nicht weiter erklärt. Den Wechsel zwischen off- und online Modus zur Kommunikation zwischen dem Ferienclubserver und dem PDA kann man über folgenden Ansätze erreichen.

Der Gast befindet sich im Funknetz des Hotels und kann Angebote einsehen und Reservierungen abschicken. Sobald der Gast diesen Bereich verlässt steht dem Gast die Anwendung nur noch im offline Modus zur Verfügung. Wenn der Gast jetzt eine Reservierung abschicken möchte, wird diese von der Anwendung auf dem Gerät in eine Warteschlage gestellt. Die Anwendung versucht nun in bestimmten Abständen sich wieder im Hotelnetz anzumelden. Sobald die Anwendung wieder eine Verbindung besitzt, sendet sie den Reservierungswunsch an den Hotelserver.

Hierbei ist es nötig, dass die Anwendung auf dem PDA immer läuft. Wenn der PDA einmal ausfällt oder die Anwendung geschlossen werden sollte, muss sicher gestellt sein, was mit den bestehenden Reservierungsanfragen passiert. Die noch nicht abgearbeiteten gehen entweder verloren oder werden beispielsweise in eine Datei gesichert. Der Benutzer muss darauf aufmerksam gemacht werden, dass solange die Anwendung geschlossen ist, es keine Möglichkeit gibt Reservierungen abzuschicken oder Bestätigungen zu erhalten. Eine einfache Lösung ist daher, ein Beenden durch den Gast zu verbieten. Allerdings bleibt die Möglichkeit des Geräteausfalls. Es kann nicht direkt entgegen gewirkt werden sondern nur probiert werden Fehler frühzeitig zu erkennen und die Daten zu sichern.

Die Anwendung benötigt für die Versendung von Reservierungen und den Abruf von Angeboten eine Funkverbindung zum Hotelnetz. Woher weiss nun die Anwendung, ob eine Verbindung besteht oder diese, wie es bei spontanen Netzverbindungen häufig vorkommt, abgebrochen ist. Die Anwendung muss die Möglichkeit besitzen diese Verbindung vor dem Aufruf der Web-Services aufzubauen oder auch auf Verbindung zu testen. Die Funkverbindung wird aber nicht direkt von der Anwendung aufgebaut sondern über das Betriebssystem. Da das Betriebssystem seine eigene Abläufe hat und auch resriktive Regeln besitzt, wer auf diese zugreifen darf, benötigt man eine Schnittstelle um von der Anwendung auf diese Funktionen zuzugreifen. Diese Schnittstelle soll Mono als Plattform der Anwendung zur Verfügung stellen. Über die Schnittstelle kann die Funkverbindung gesteuert werden.

Zusammenfassung

Die Synchronisation wird vereinfacht indem gesagt wird, dass die aktuellen Angebote immer beim Anbieter zu finden sind. Diese werden einmal pro Tag vom Hotel abgeglichen und als “Grundgerüst” den Gästen in einer Datenbank zur Verfügung gestellt. Der PDA überschreibt die Angebote mit denen auf dem Hotelserver und speichert diese in einer XML Datei ab, wo spezielle Informationen zu dem Angebot noch einmal extra gekennzeichnet werden.

Die Steuerung zwischen off- und online Modus auf dem PDA wird von der Anwendung gesteuert, die mit Hilfe von Mono auf die Funknetzeinstellungen des Betriebssystemes zugreift. Um eine Vereinfachung der Bedienung des Gerätes zu besitzen, wird die Anwendung so gestartet, dass sie sich nicht durch den Gast beenden läßt.

In der Realisierung soll getestet werden inwieweit sich diese Lösung als praktikabel herausstellt und Mono die Möglichkeiten bietet, die einzelnen genannten Punkte zu erfüllen.

5. LaborUmgebung

In diesem Kapitel soll ein Überblick darüber gegeben werden, welche Techniken für die Realisierung benutzt werden und was diese darstellen und bedeuten. Es wird die Idee von Web-Services erläutert, eine kurzer Einblick in die Geschichte und eine Erläuterung zu den wichtigsten Bestandteilen von Web-Services gegeben. Im Weiteren gibt es eine Erklärung was Mono eigentlich ist und in welchen Verhältnis es zu .NET steht. Zusätzlich zu den zu untersuchenden Techniken von Mono als Plattform für Web-Services kommt eine Beschreibung des mobilen Endgeräts von HP und des darauf eingesetzten Betriebssystems Linux. Mit der Darstellung der schnurlosen Netzwerkschnittstelle auf dem Endgerät wird die Beschreibung der Laborumgebung abgeschlossen.

5.1. Webservices

Eine immer wieder auftauchende Vision eines “programmierbaren Internets”, in der alle Arten von Anwendungen und Geräten ihre Fähigkeiten und Funktionalitäten problemlos miteinander teilen und zur Verfügung stellen, könnte mit Hilfe von Web-Services jetzt realisiert werden. So lässt sich auch der “Hype” um diese/n Technik/Dienst in den letzten Jahren erklären [Knuth 2003].
Eine recht einfache Erklärung zu Web-Services kommt aus der IX [Wirdemann 2004]:

> Ein Web Service ist ein Dienst, dessen Funktionen der Anwender über das Web nutzt. Die Kommunikation erfolgt mittels über HTTP(S) transportierte XML-Nachrichten. Primäre Aufgabe der Integrationstechnik ist die Kopplung proprietärer Systeme, denen sie neue Zugangswege verschafft.

Geschichte von Web-Services

In der IT-Geschichte gab es schon mehrere Ansätze zur Realisierung der beschriebenen Vision 5.1. Ein erster Ansatz Ende der achtziger Jahre war dabei DCE - Distributed Computing Environment. Die Idee einer einheitlichen API auf allen Plattformen (Betriebssystemen), von Großrechnern bis zum kleinen 16-bit-Windows-System, war aus verschiedenen Gründen

(viele davon firmenpolitisch motiviert) damals nicht erfolgreich. Aber auf Grund der Veränderung in den Informationssystemen von einem einzigen großen Datencenter hin zu vielen kleinen unterschiedlichen Systemen erlangte die Systemintegration eine immer grössere Bedeutung und es wurde vermehrt nach einer Standardisierung der Netzwerktechnologien gesucht. Aus dem Zusammenschluss verschiedener Firmen (Object Management Group (OMG)) zur Schaffung von Standards bei Verteilten Systemen entstand Common Object Request Broker Architecture (CORBA). Die Initiative von CORBA wetteifert seit der frühen 90er Jahren mit der Technologie von Microsoft: COM (Component Object Model Architecture). Leider haben diese Technologien einige Nachteile, wie z. B. eine schwierige Skalierbarkeit, eine hohe Komplexität und ein fehlender universeller Standard zur Datenrepräsentation, die den Einsatz im Internet erschweren und zum Teil auch unmöglich machen. Die bis dahin entstanden Forderung lauten:

- Internationalisierung
- Fehlertoleranz bei Netzwerkproblemen
- gute Unterstützung für Anwendungsentwicklung und Workflowverwaltung
- einfache Handhabung in Verbindung mit Transaktionen, Sicherheit und Organisation
- möglicher Datenaustausch zwischen verschiedenen Unternehmen und Systemen
- Skalierbarkeit von heterogenen Netzwerken

Mitte der 90er Jahre entwickelte Hewlett Packard (HP) einen Standard zum Datenaustausch mit Hilfe von HTTP und XML der “e-Service” genannt wurde. Durch die UserLand community, unter Leitung von Dave Winer, wurde der Standard von XML-RPC entwickelt. Ein sehr einfaches Protokoll für Funktionsaufrufe auf entfernten Rechnern (Servern). 1999 wurde dann Simple Object Access Protocol (SOAP) veröffentlicht, welches wieder in einem Zusammenschluss mehrerer grosser Firmen entstand. SOAP definiert dabei das Standardprotokoll zur Kommunikation mit Web-Services. Durch die Entwicklung von Web Services Description Language (WSDL) und Universal Description, Discovery and Integration (UDDI) durch IBM, Microsoft und Ariba im Jahre 2000 war der Grundstein für Web-Services gelegt. Auf dem “W3C Workshop on Web services” im April 2001 wurden weitere Standards festgelegt und Arbeitsgruppen für die Entwicklung und Betreuung gegründet. Seit 2002 sind Web-Services eine vollständige Technologie, um verschiedene Systeme miteinander kommunizieren zu lassen [Ogbuji 2003].

Was sind Webservices?

Was sind Web-Services und welche Eigenschaften besitzen sie? An dieser Stelle soll eine kurze Zusammenfassung der Eigenschaften von Web-Services gegeben werden:

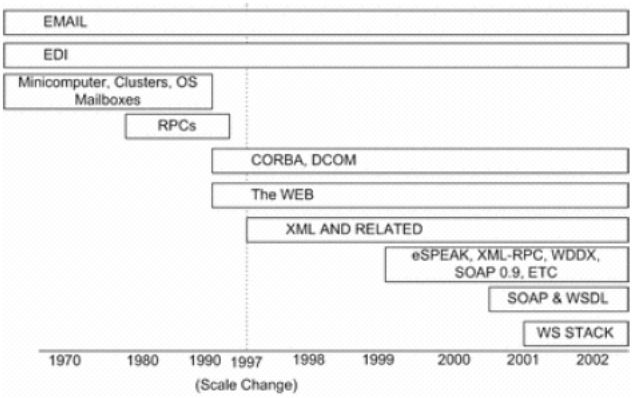

Abbildung 5.1.: Entwicklung von Web-Service

- Ein Web-Service ist über das Internet verfügbar. Web-Services kommunizieren plattformunabhängig und benutzen ein sprachunabhängiges Internetprotokoll. Diese Protokolle sind einfach in heterogene Umgebungen integrierbar.
- Web-Services bieten Schnittstellen, die es ermöglichen sie aus anderen Programmen heraus aufzurufen. Dieses “application-to-application programming interface” kann von jeder Anwendung oder jedem Dienst aufgerufen und benutzt werden. Diese Schnittstellen verbinden eine mögliche logische Anwendung mit der Funktionalität der Web-Services.
- Die Web-Services sind im Internet registriert und ermöglichen Benutzern nach bestimmten Diensten zu suchen und diese dann zu nutzen.
- Web-Services bieten eine lose Kopplung zwischen einzelnen Systemen. Sie kommunizieren über den Austausch von Nachrichten miteinander und ermöglichen mit Hilfe der Schnittstellen eine abstrakte Netzwerkschicht im System, mit der sich Verbindungen einfach einbinden und flexibel gestalten lassen.

Web-Services können in jeder Programmiersprache und auf jedem Betriebssystem entwickelt und benutzt werden. Ermöglicht wird dies durch die Extensible Markup Language (XML), die Web-Services als Sprache benutzt. Web Services benutzen XML um die eigenen Schnittstellen zu beschreiben und ihre Nachrichten zu “verpacken”. Der Nachrichtenausstausch bei Web-Service findet über XML basierte Schnittstellen und XML basierte Nachrichten statt. Die dabei verwendeten Protokolle sind z. B. http oder smtp. Der Vorteil dabei ist, dass jede Anwendung XML unterstützt und interpretieren kann. Hinzu kommt, dass

mit http oder smtp keine speziellen Protokolle zum Nachrichtenaustausch benutzt und diese somit überall vorhanden sind und unterstützt werden.

XML ist aber nicht die einzige Technik mit der Web-Service arbeiten. Die Technologien, auf denen Web-Services beruhen, benutzen lediglich XML. Die drei entscheidenden Punkte sind:

- Simple Object Access Protocol (SOAP); das Standardprotokoll zur Kommunikation
- Web Services Description Language (WSDL), die Beschreibungssprache von Web-Service. Dort werden die einzelnen aufrufbaren Funktionen beschrieben und dargestellt.
- Universal Description, Discovery and Integration (UDDI) wird benutzt um die einzelnen Web-Services zu registrieren und sie im Internet für Benutzer auffindbar zu machen.

In der Grafik 5.2 sind die einzelnen Bestandteile in ihrem Zusammenhang und die Abläufe beim Benutzen von Web-Services dargestellt. Wenn ein Anbieter seine Services verschiedenen Benutzer bereitstellen will, definiert er diese mit Hilfe von WSDL und registriert diese Services bei einer UDDI-Registrierungstelle im Internet. An dieser Stelle wird dann auf die WSDL Beschreibung verwiesen. Somit steht der Service zur Verfügung.
Möchte ein Benutzer jetzt diesen speziellen Service nutzen, sucht er bei der UDDI Registierungsstelle nach diesem Service. Wenn er erfolgreich ist, bekommt er die WSDL Beschreibung und den Verweis, wo sich der Service befindet. Der Anwender benutzt die WSDL Beschreibung zum Bau einer SOAP Nachricht, über die er dann mit dem Service kommunizieren kann und diesen benutzt [systinet:2003 2003].

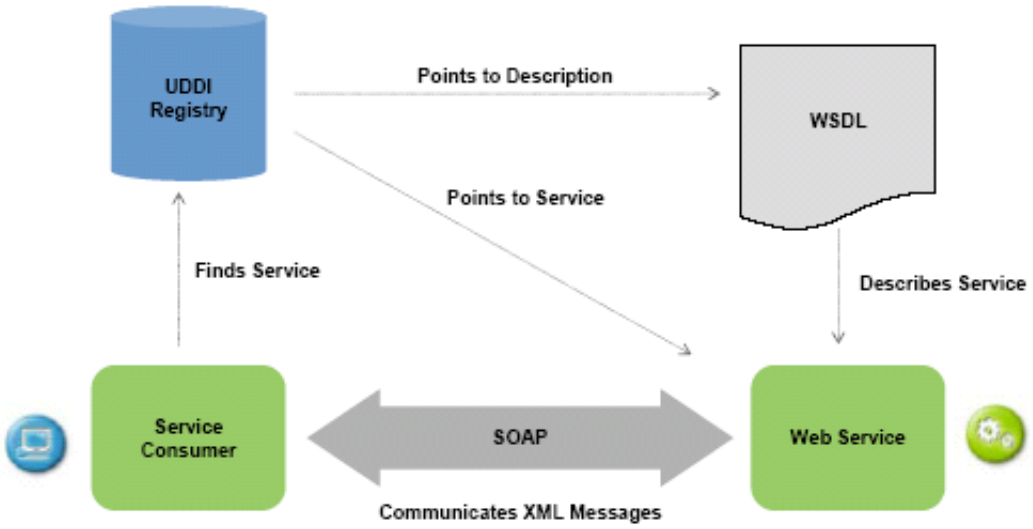

Abbildung 5.2.: Die wichtigsten Bestandteile von Web-Services: SOAP, WSDL und UDDI

Web Services Technologien

Auf eine genauere Beschreibung der einzelnen Technologien soll an dieser Stelle verzichtet werden. Für eine ausführliche Dokumentation wird auf die zitierten Quellen verwiesen. SOAP ist ein erweitertes XML Nachrichten Protokoll, das die Basis für Web-Services bildet. SOAP bietet einfache Mechanismen zum Austausch von XML-Nachrichten zwischen zwei Anwendungen. Die Dokumentation befindet sich auf den Internetseiten von W3C [Soap:2003 2003]. WSDL ist eine XML basierte Beschreibung der einzelnen Web-Services. Es beschreibt, welche Funktionalität der Web-Service besitzt, wie es kommuniziert und wie man darauf zugreift. SOAP Entwicklungstools können aus dieser Beschreibung eine direkte Umsetzung in eine SOAP Schnittstelle realisieren, die in der Anwendung zum Einsatz kommt. Ein WSDL Dokument besteht hauptsächlich aus fünf Definitionen:

- Types: Beschreibt welche Datentypen in der Nachricht benutzt werden [Schema:2004 2004].
- Messages: Beschreibt das Format der Nachricht, welche Ein- und Ausgaben möglich sind.
- Port Type: Definiert eine Anzahl von Operationen. Beim Einsatz von SOAP-RPC repräsentiert jede Operation eine Methode.
- Binding: Bindet die in Port Type beschriebenen Operationen an ein konkretes Protokoll und Datenformat.
- Service: Definiert eine Menge von Ports und macht sie über einen Zugangspunkt (URL) verfügbar.

UDDI ist ein Mechanismus zur Registrierung und Kategorisierung von Web-Services. Es bietet die verschiedenen Services an und stellt den Ort zur Verfügung, an dem diese sich befinden. UDDI ist dabei selbst ein Web-Service bei dem die Benutzer ebenfalls über SOAP Nachrichten kommunizieren. Die UDDI Standards werden von OASIS verwaltet und weiterentwickelt. Es gibt zwei verschiedene Möglichkeiten der Registrierung: zum einen die Private Regestrie, bei der jeder alle Dienste zur Verfügung stellen kann und jeder diese benutzen darf. Die zweite Möglichkeit ist die B2B Registrie (“Business to Business”), bei der Firmen, die über Web-Services kommunizieren, ihre Dienste anmelden können und mehr Kontrolle darüber erhalten, wer dieses Angebot nutzen darf.

5.2. .NET - Mono

.NET, was ist das? Was macht diese Plattform von Microsoft interessant und was ist neu an dieser Technologie? Microsoft beschreibt .NET auf seiner Webseite ([.net:2004 2004]) wie folgt:

> Bei Microsoft .NET handelt es sich um einen Satz von Softwaretechnologien zum Verknüpfen von Informationen, Menschen, Systemen und Geräten. Diese Technologie der nächsten Generation basiert auf Webservices - kleinen Anwendungsmodulen, die über das Internet sowohl miteinander als auch mit anderen, größeren Anwendungen kommunizieren können.

Das heißt, Microsoft bietet eine sprachunabhängige Plattform an, die es ermöglicht, die verschiedenen Programmiersprachen zu verbinden und sich für bestehende Schwierigkeiten die geeignete Technik zur Problembehandlung heraus zu suchen. Um den oben aufgeführten Anforderungen gerecht zu werden, bietet Microsoft drei verschiedene übergreifende Ansätze der Umsetzung an. Dabei handelt es um die sogenannten Web-Services, .NET Remoting Services für verteilte Anwendung im Lan und die .NET Enterprise Services. Letztere können als Konkurrenzprodukt zu der Technologie J2EE von Sun angesehen werden.

Die Common Language Runtime (CLR) ist der wichtigste Teil von .NET. Die CLR ist ein Just-in-Time Compiler mit dem .NET das Programm in Maschinensprache übersetzt 5.3. Es ist vergleichbar mit der virtuellen Maschine von Java.

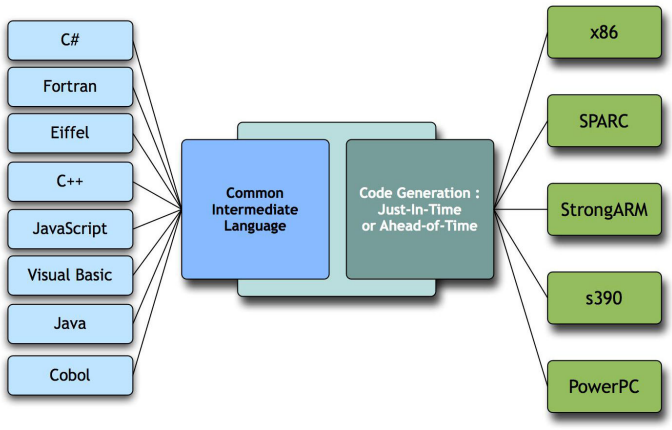

Abbildung 5.3.: Von der Programmiersprache zur Maschine [de Icaza 2005]

Die CLR führt den standardisierten Zwischencode der Common Intermediate Language (CIL) aus. Die CIL hieß früher Microsoft Intermediate Language (MSIL), wurde aber im Rahmen der Standardisierung durch die ECMA umbenannt. Für die CIL wurde ein sprachübergreifendes System von objektbasierten Datentypen definiert, so dass auch in unterschied-

lichen Sprachen geschriebene Programmteile auf gemeinsame Ressourcen zugreifen können ([Wiki.net:2004 2004]). Abbildung 5.4 ist eine Darstellung der CLR und deren einzelner Komponenten.

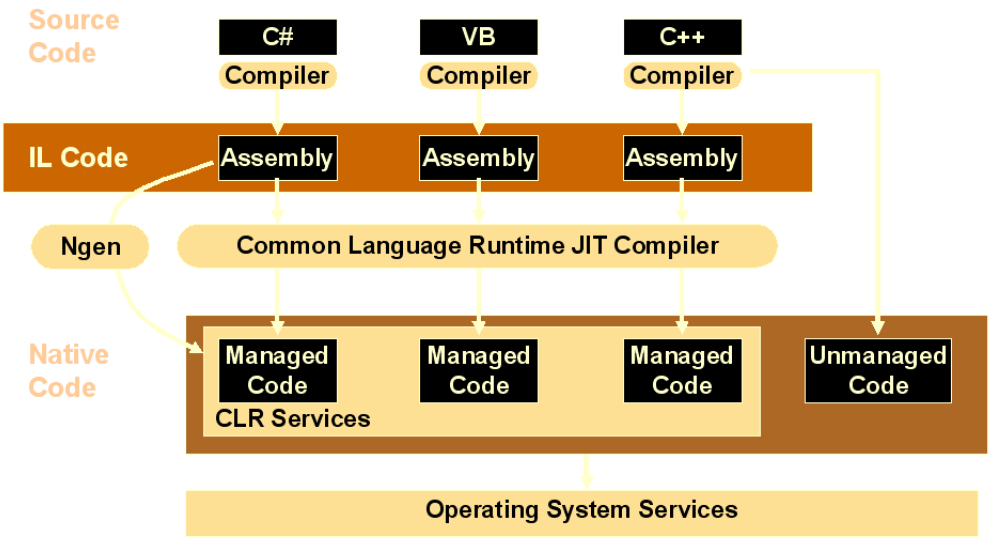

Abbildung 5.4.: .NET, Common Language Runtime [Kulicke 2005]

Im Zuge der Standardisierung von CLR und CIL durch die ECMA wurde es möglich .NET auch anderen Plattformen zugänglich zu machen. Im Moment gibt es in diesem Zusammenhang mehrere Projekte:

- Mono [mono-project:2004 2004]
- Dotgnu [dotgnu:2004 2004]
- OCL [ocl:2004 2004]

Hierbei scheint Mono im Moment das interessanteste Projekt zu sein, da es im Vergleich zu den anderen Projekten eine vollständige Unterstützung von ADO-Net (Datenbankzugriff) und ASP (ActiveSererPages) bietet. Mono ist eine “Open Source” Implementierung der Microsoft .Net Plattform mit Werkzeugen, Biblioteken und Compilern, die es ermöglichen Anwendungen auf verschiedenen Plattformen wie Linux, MacOS X, Solaris, BSD und Microsoft sowie verschiedenen Programmiersprachen zu entwickeln und zu verwenden.

Mono ist keine komplette Umsetzung von .Net. Es fehlen Elemente wie `Windows.Forms` und `Visual Basic`. Erwähnenswert ist, dass in den nächsten Versionen von Mono die fehlenden Elemente hinzukommen sollen. Es wurden bei der aktuellen Version jedoch zusätzliche Datenbankschnittstellen, verbesserte XML-Schnittstellen, LDAP-Funktionalität sowie mit GTK# eine Umgebung für graphische Schnittstellen mitgeliefert, die .Net erweitert und verbessert.

Mono versucht kompatibel zu Microsoft .Net zu bleiben. Somit ist es möglich mit Hilfe von Mono Anwendungen zu entwickeln, die frei verfügbar sind und unter Windows keine Lizenzprobleme mit sich bringen. Die Portierung von .Net Programmen unter Windows auf andere Plattformen wie Linux wird erleichtert. Mono entstand 2001 im Zusammenhang mit dem GNU Projekt Gnome, bei dem es ähnliche Schwierigkeiten mit “gewachsenen” Programmierschnittstellen gab wie unter Windows. Die Idee der Sprachunabhängigkeit war unter Linux ebenso attraktiv wie unter Windows. Des Weiteren bietet sich dadurch die Möglichkeit unter Windows entwickelte Programme auch unter Linux zu nutzen, was die Attraktivität von Gnome und Linux steigert.

Was bietet Mono? Mono ist wie .Net plattformunabhängig und vereint mehrere Programmiersprachen unter sich, wie C#, Java, VB.NET, MONOLOGO, ASP und Web-Services (SOAP).

In der Abbildung 5.5 von sind die einzelnen Bestandteile von Mono und deren Zusammenhang dargestellt.

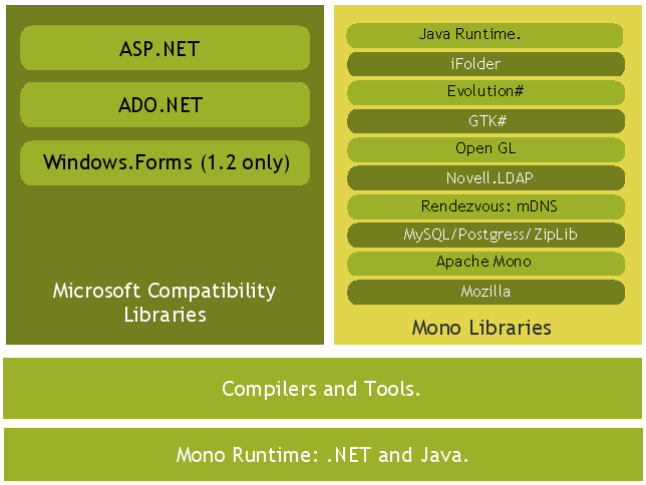

Abbildung 5.5.: Mono Laufzeitumgebung [de Icaza 2005]

Die Mono-Runtime bietet drei Möglichkeiten der Code-Ausführung.

- Just-in-Time, der Bytecode (Common Intermediate Language, CIL) wird vor der Ausführung in Maschinencode ausgeführt [CIL:2002 2002].
- Ahead-of-Time
  - Basiert auf der Just-in-Time Ausführung.
  - Der Maschinencode wird aber im Voraus erzeugt.
  - Senkt die Startup-Kosten.

  - Erlaubt mehr Optimierungen.
  - Die im AOT-Schritt erzeugten Binärdateien sind nicht mehr Plattform-unabhängig!
- mint, der Mono-Interpreter interpretiert den Bytecode(CIL) in Echtzeit während das Programm ausgeführt wird

5.3. Linuxbasierter PDA

In den vorherigen Kapiteln war die Rede von einem mobilen Endgerät oder, in dieser Untersuchung, einem Personal Digital Assistent (PDA) mit dem es dem Gast möglich sein soll sich in dem Informationssystem des Ferienclubs anzumelden und mit Hilfe einer installierten Anwendung die Angebote zu buchen. Ein PDA bietet eine hohe Mobilität. Durch sein geringes Gewicht und die geringe Grösse kann dieser überall bei sich getragen und jederzeit genutzt werden. Durch einen Batteriebetrieb ist es möglich dieses Gerät mehrere Stunden ohne eine stationären Stromversorgung auskommen.

Ipaq Pocket PC

Das Gerät, das bei dieser Untersuchung benutzt wurde, ist ein Ipaq Pocket PC 3870 der Firma Compaq/HP. Das Gerät verfügt über die grundsätzlichen Funktionen, die bei dieser Untersuchung verlangt werden. Dazu zählen ein Funkmodul mit dem die Verbindung zum Hotelnetz aufgebaut werden kann, ein farbiges TFT-Display mit einer Auflösung von 240 x 320 Pixel und 64-K-Farbunterstützung bei einer 0,24-Lochmaske. Das Display besitzt eine automatische Helligkeitsregelung, um es der jeweiligen Umgebung anzupassen. Das Gerät besitzt ein 32-bit Intel StrongARM SA-1110 Mikroprozessor mit 32 MB ROM (Flash ROM) und 64 MB RAM. Zusätzlich sind in das Gerät Lautsprecher und Mikrophon eingebaut. Das Gerät hat eine Batterielaufzeit von ca. 14 Stunden, wodurch der Gast das Gerät tagsüber bei jeglichen Aktivitäten bei sich führen kann 5.6.

Linux auf dem Ipaq

Das auf dem Ipaq vorinstallierte Windows CE wurde aus verschiedenen Gründen durch ein Linux Betriebssystem ersetzt. Wegen Sicherheitsbedenken wird auf dem Gerät ein Mehrbenutzerbetrieb benötigt [Lüpke 2004],[Mählmann 2004]. Windows CE bietet nur einen Single-User Modus an. Die Bluetooth-Schnittstelle erlaubt auf einem Linuxbetriebssystem eine bessere und feinere Konfiguration im Vergleich zu Windows CE. Der wesentliche Punkt für die

Abbildung 5.6.: Ipaq Pocket PC 3870

Verwendung von Linux ist darin zu suchen, dass Mono im Moment nur für einen linuxbasierten PDA erhältlich ist. Weil auf dem Gerät auch verschiedenste Komponenten von Mono untersucht werden sollen, kann kein Windows CE mit .NET eingesetzt werden.

Verwendung findet die Linuxvariante von der Internetseite Handhelds.org [Handheld:2005 2005]. Weitere Informationen zu dem PDA und Linux befinden sich auf der Internetseite von Ipaqlinux [Ipaq:2005 2005]. Das Familiar Project ist eine Linux-Distribution für PDAs. Sie entstand im Jahr 2000. Das Projekt besteht aus mehreren unabhängigen Entwicklern, die sich zum Ziel gesetzt haben, Linux auf dem PDA als Betriebssystem zu integrieren. Die aktuelle Version 0.81 besteht aus verschiedenen Komponenten: Als graphische Oberfläche können, wie bei Desktop Linux, verschiedene Module benutzt werden. Es wird unterschieden zwischen GPE [GPE:2005 2005]5.7 und Opie [OPIE:2005 2005]5.8. GPE benutzt dabei die GTK-Bibliotheken, die auch schon von Gnome [Gnome:2005 2005] her bekannt ist. Bei Opie kommt die QT-Bibliothek von TrollTech(KDE)[Kde:2005 2005] zum Einsatz. Besonders wichtig ist die Anlehnung an die Microsoft Windows Umgebung 5.7,5.8, um somit eine gute Ergonomie zu bieten.

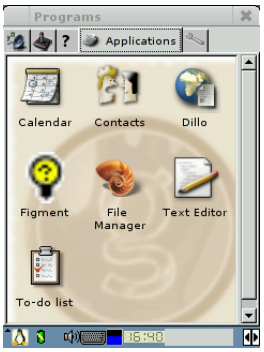

Abbildung 5.7.: GPE auf dem IPAQ

Abbildung 5.8.: Opie auf dem IPAQ

Weiterhin werden Werkzeuge, wie z. B. grep und vi aus der Busybox [Bbox:2005 2005] benutzt, um Ressourcen zu schonen. Zur Unterstützung der Hardware und der einzelnen Anwendungen wird das Paketsystem ipkg unterstützt. Bei ipkg [Ipkg:2005 2005] handelt es sich um eine an dem Debian Paketsystem dpkg angelehnte Variante zum Installieren von Updates und neuer Software. Mit ipkg erzielt man die größtmögliche Anpassung an die Anforderungen des PDA, da man die Software die benötigt wird individuell installieren kann. In dieser Untersuchung sind die Bluetooth Pakete von Bedeutung.

Zur Installation von Linux möchte ich auf die sehr gute Dokumentation auf der Internetseite von familiar.handheld.org verweisen. Dort ist ausgehend von einem Windows CE Betriebssystem erklärt, wie in den einzelnen Schritten der Installation vorzugehen ist [Gettys 2005].

Die Pakete zur Installation von Mono auf dem PDA kommen von der Ipkg-Paketverwaltung im Internet [MonoHandhelds:2004 2004].

Bluetooth

Die Möglichkeiten zum Datenaustausch bestehen aus IrDA (Infrared Data Association) und Bluetooth. Mit der eingebauten Infrarotschnittstelle können Daten aus einer PIM-Anwendung [1] mit weiteren Geräten synchronisiert oder getauscht werden.
Der wesentlich interessantere Anteil ist das eingebaute Bluetooth-Funkmodul, welches bei dem Ipaq standardmäßig mitgeliefert wird. Auf dem Hotelserver wird ein Acer Bluetooth USB Dongle mit einer Reichweite von 20 Metern und einer maximalen Übertragungsrate von 10 Mbit/s 5.9 verwendet.

Die Vorteile von Bluetooth sind die geringen Produktionskosten, niedriger Energieverbrauch (z. B. Batteriebetrieb), die breite Unterstützung vieler Firmen und die wachsende Verbreitung dieser Technologie. Das bedeutet, dass sich Bluetooth in den verschiedensten Geräten

[1] PIM (Personal Information Management), Verwaltung von Adressen, Terminen und Notizen

Abbildung 5.9.: Acer Bluetooth USB Doongle

wie z. B. Computer, Handy, PDA oder in Peripheriegeräten wie Mäusen und Tastaturen wieder findet. Die Wireless Technologie wurde ursprünglich von Ericsson entwickelt und wird inzwischen von einem Konsortium mehrerer Hersteller, der Bluetooth SIG (Special Interest Group)[Sig:2005 2005], koordiniert. Es ist einfach verschiedenste Geräte in einer Entfernung von ca. 10 Metern bis zu 100 Metern (je nach verwendeten Bluetooth Chipsatz) miteinander zu verbinden. Zur Zeit gibt es 13 verschiedene Profile, die spezifiziert wurden um unterschiedliche Standardaufgaben zu erfüllen. Diese können wiederum auf unterschiedlichen Protokollen aufgebaut sein. Ein Beispiel dafür wäre die Datenübertragung mittels dem Profil File Transfer (FP) oder die Benutzung des lokalen Netzwerkes (Lan Access, LAP)[Hilzinger 2004].

Bluetooth ist ein Funknetz mit sehr geringer Ausdehnung (Piconetz). Eine Picozelle enthält einen Master und maximal sieben Slaves. Der Master wird in dieser Umgebung vom Hotelserver dargestellt, der als Gateway für die Nutzung des Hotelnetzs und der Angebote dienen soll. Der einzelne PDA meldet sich am Master an und baut dabei eine Netzwerkverbindung zu dem Hotelserver auf.

Für den Einsatz von Bluetooth als Netzwerkverbindung gibt es zwei Möglichkeiten der Realisierung. Die erste Möglichkeit ist über Dial Up Networking (DUN) und die zweite Möglichkeit ist Personal Area Network (PAN). Für die Untersuchung ist es hierbei nur wichtig eine TCP/IP Verbindung zur besitzen, um die Web-Service zu untersuchen. Im Anhang 5.3 wird genauer erläutert, wie Bluetooth als Netzwerk eingerichtet werden muss.

5.4. Versuchsaufbau

Die Beschreibung der Umgebung soll nur dem groben Verständnis dienen und ein Gefühl vermitteln, wie es hier getestet wird und darüber hinaus nicht weiter vertieft werden.

Wie im Entwurf bereits besprochen, werden mehrere Rechner benötigt; dabei dienen zwei als Server und einer als mobiler Client. Ein Server repräsentiert den Anbieter.Der zweite dient gleichzeitig als Hotelserver und Client, der die Angebote bei den Anbieter abholt. Dieser Rechner benötigt zwei Kommunikationsschnittstellen. Eine einfache Netzwerkverbindung zum Internet und eine zweite in Form eines Funkmoduls für die Übertragung zu dem mobilen Endgerät. Das mobile Endgerät soll hierbei ein PDA sein, der für den Gast einfach zu bedienen ist und ihn in seiner Mobilität nicht einschränkt. Im Folgendem folgt eine kurze Darstellung der einzelnen Geräte:

- **Anbieter:** Besteht aus einem handelsüblichen Computer, der im Internet über eine Internetadresse erreichbar ist. Dieser Server wird als Webserver arbeiten. Als Betriebssystem wird Debian in der aktuellen “sarge” Distribution benutzt. Der Webserver wird mit Hilfe von Apache2 realisiert. Mono wird in der Version 1.05 zum Einsatz kommen (nur in “sid” verfügbar). Mono wird dabei über das Apachemodul in den Webserver eingebunden. Der Server wird nur die Aufrufe vom Hotelserver bearbeiten.
- **Hotelserver:** Besteht ebenfalls aus einem “normalen” Computer der über eine Bluetoothschnittstelle verfügt, um mit dem PDA zu kommunizieren und über eine Netzwerkkarte für die Verbindung in das Internet. Das Betriebssystem ist ebenfalls ein Debian “sarge” und benutzt Mono in der Version 1.05. Bei dem Hotelserver wird der eigene Webserver XSP eingesetzt um einen Vergleich zu dem Apachemodul zu erhalten. Die Datenbank wird von MySql gestellt. Die Bluetoothschnittstelle wird wie auf dem PDA realisiert und benutzt die Bluetooth-utilz.
- **Client:** Als Client dient der HP mit dem Linux, das im vorherigen Kapitel beschrieben wurde.

Entwicklungsumgebung

Um Mono entwickeln zu können, sollte für die Programmierung eine bedienbare Entwicklungsumgebung mit dem Funktionsumfang von beispielsweise Microsofts Visuell Studio vorhanden sein. In dieser Untersuchung werden drei verschiedene Umgebungen eingesetzt.

Eine Entwicklungsumgebung für Mono unter Windows ist der unter GNU Lizenz stehende SharpDevelop von der Firma icsharpcode.net. Die Umgebung ist komplett in csharp programmiert und ist für .NET als auch Mono geeignet. Es besitzt alle Anforderungen, die an eine Entwicklungsumgebung gestellt werden, wie z. B. Syntax Highlighting, Code Completion oder Unit- Testumgebung A.2.

Die freie Entwicklungumgebung Monodevelop (GNU Lizens) entstand im Rahmen von Mono. Sie befindet sich noch in keinem stabilen Status. Der Aufbau und die Bedienung sind ähnlich der von SharpDevelop A.3.

Es werden noch weitere Umgebungen für Mono angeboten. Zwei interessante Projekte sind dabei das Plugin für Eclipse, was aber noch nicht den Funktionsumfang bietet wie die beiden vorher beschriebenen. Der Entwickler Brad Merill stellt eine Umgebung für csharp unter dem Emacs zur Verfügung. Diese funktioniert ähnlich wie JDE für Java.

Für die Entwicklung und Umsetzung von einer graphischen Beutzeroberfläche wird in dieser Untersuchung mit GLADE 2 entwickelt. Glade 2 kommt ursprünglich aus dem GNOME Entwicklungsbereich und wurde um eine c# Bibilothek erweitert, mit der es möglich ist eine GUI einfach und schnell zu bauen. Es ist ebenfalls unter der GNU Lizenz erhältlich. Es ist einsetzbar unter Linux, wie auch unter den verschiedenen Windowsplattformen A.4.

6. Prototyp/Realisierung

Dieses Kapitel dient zum Testen von Mono als Plattform für den in den vorherigen Kapiteln vorgestellten Anforderungs- und Designentwurf. Es soll eine Anwendung getestet werden, die dem vorgestellten Entwurf möglichst nahe kommt, um einen realistischen Einblick in die Möglichkeiten von Mono zu bekommen. Ein wichtiger Aspekt ist dabei die Entwicklung von Web-Services und die Benutzung der selben auf mobilen Endgeräten, wie den in dem vorherigen Kapitel vorgestellten IPAQ von HP. Zusätzlich soll getestet werden, wie andere grundlegende Funktionalitäten, wie z. B. in die Datei schreiben, Datei einlesen, XML Dokumente bearbeiten und ein Graphisches User Interface entwickeln, möglich und einsetzbar sind.

Es wird erläutert, wie die Realisierung aussehen wird und welche Teile aus dem Entwurf nicht mit implementiert wurden. Nach der Erläuterung der Zielsetzung des Prototypen steht dann die konkrete Implementierung der einzelnen Web-Services und die Testanwendung für den Client. Dazu werden einzelne Bestandteile, die speziell für den Prototypen sind, hervorgehoben und genauer erläutert. Es werden einzelne “Codezeilen” mit eingebunden, die einen Eindruck vermitteln sollen, wie man mit Mono und der Programmiersprache C# einzelne Anwendungen programmiert und entwickeln kann. Zum Abschluss steht eine Bewertung des Prototypen und somit der Möglichkeiten, Mono als ernsthafte Plattform für die Entwicklung von Web-Servicen einzusetzen.

6.1. Umsetzung des Entwurfes

An dieser Stelle soll nun die Umsetzung der Entwurfsidee folgen. Dabei werden die Web-Services, wie im Entwurf beschrieben, übernommen und sollen dementsprechend implementiert werden. Bei der Implementierung wird das Proxy-Pattern [Gamma Erich und Vlissides 1995] benutzt werden. Der Anbieter wird in der Realisierung durch die im Entwurf 6.1 besprochenen Autovermietung dargestellt. Die dabei verwendete Struktur soll in den Prototypen mit übernommen werden.

Auf dem Hotelserver werden die Web-Services implementiert und eine Anwendung geschrieben, die mit dem Anbieter kommuniziert und dessen Informationen abspeichert. Als Grundsätzliche Struktur wird das Klassendiagramm aus dem Entwurf zum Einsatz kommen.

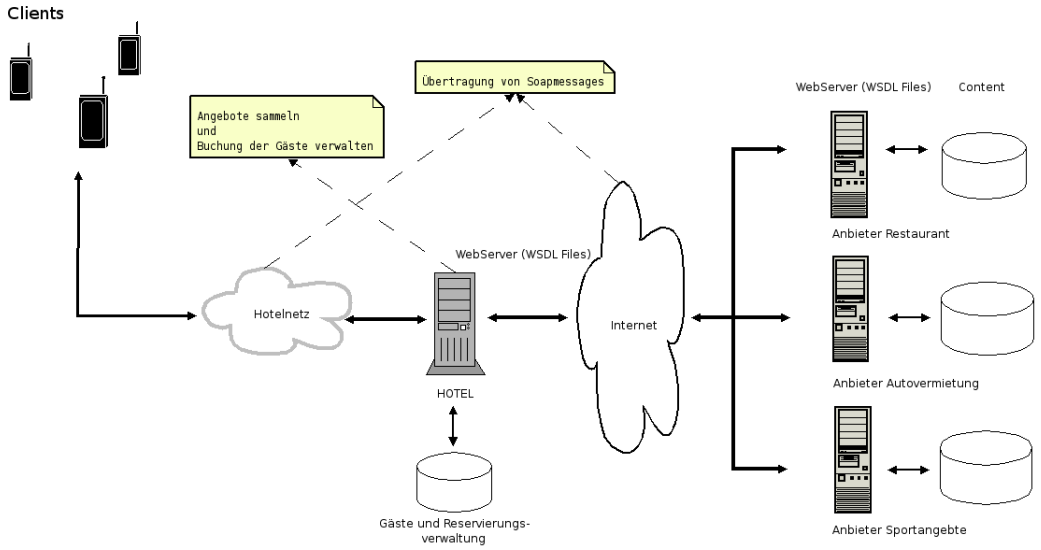

Abbildung 6.1.: Komponentendarstellung des Ferienclubs

Auf dem Client wird eine Anwendung mit einer Graphischen Oberfläche aufgebaut, über die es möglich sein wird, mit Hilfe der Web-Services mit dem Hotel zu interagieren und sich Angebote anzusehen und abzuspeichern. Für das Netzwerk zwischen Hotelserver und PDA wird eine Bluetoothverbindung aufgebaut.

Versuchsaufbau

Bei der Realisierung wird auf verschiedene Teile des Entwurfes verzichtet, da diese nicht zum Testen der Applikation dienlich sind, sondern eher einen sauberen Softwareentwurf betreffen. Im Mittelpunkt dieses Prototypen soll die Plattform Mono und deren Funktionalität stehen.
Bei der Realisierung des Anbieters wird darauf verzichtet verschiedene Anbieter darzustellen. Es wird in einer Datenklasse ein “Angebot” bereitgestellt. Bei dem Hotel wird das Angebot in einer Textdatei oder Datenbank abgelegt, ohne eine “echte” Reservierungsanwendung einzuführen. Der Client wird zu den einzelnen Aufrufen für die Web-Services nur über die zusätzliche Funktion des Abspeicherns in einer Datei verfügen. Die Problematik der Synchronisation, der Funkvernetzung (oftmals Verbindungsabbruch), sowie der Sicherheit wird hier vernachlässigt. Der Ablauf aus dem Sequenzdiagramm im Kapitel 4 soll implemetiert werden. Dafür werden folgende Punkte in den nachfolgenden Kapiteln bearbeitet:

- Implementierung der Web-Services auf den Webservern.
- Die WSDL Schnittstellen erzeugen.

- Die minimalen Anwendungen auf dem Client und dem Hotelserver entwickeln.

Bei der folgenden Implementation sind verschiedene Programmierbeispiele aus dem Buch von [Dumbill und Bornstein 2004] und [Ferrara und MacDonal 2003] und weiterhin von der Webseiten des Monoprojektes [mono-project:2004 2004], [Mono-book:2004 2004] sowie [.netProg:2005 2005] verwendet worden.

Mobiles Endgerät

Auf dem PDA werden folgende Methoden implementiert, die zur Benutzung der Web-Services dienen werden.

- GetBargains()
- bookABargain(Bargain aBargain)
- SubmitAReservation(Reservation reservation)
- GetStatus(String reservationNumber)

Die Implementation dieser Methoden, die die Web-Services auf dem Hotelserver aufrufen, bieten keine grösseren Schwierigkeiten. Mono muss an dieser Stelle nur die Möglichkeit eines Web-Service Clients bieten, um die Requests in SOAP Nachrichten zu verpacken, abzuschicken und die Antworten des Servers wieder zu entpacken. Die dafür vorgesehenen Bibliotheken sind als Namespace in Mono enthalten und müssen nur eingebunden werden. Von besonderen Interesse ist die GUI, die dem Gast die Angebote zugänglich machen soll. Mono bietet in diesem Zusammenhang noch nicht die graphischen Bibliotheken, wie .NET, sondern nutzt die Bibliotheken von GTK. Es muss getestet werden, wie gut sich diese auf dem PDA einsetzen lassen, wie groß der Funktionsumfang der Bibliotheken ist und ob sie eine Alternative zu den Windows Bibliotheken darstellen.

Im Folgenden wurde eine kleine GUI-Anwendung mit der GTK-Bibliothek gebaut 6.2.

Der dabei entstandene Code ist relativ verständlich und einfach aufgebaut.

```
1  using System;
   using System.Collections;
3  using Gtk;

5  namespace dip.client
   {
7      class ClientGui {
           private static Entry firstname_entry, lastname_entry,
9              email_entry, hotel_entry, datumVon_entry, datumBis_entry,
                   autoTyp_entry, cancel_entry, send_entry;
```

Abbildung 6.2.: GUI mit GTK

```
11          public void Start ()
            {
13              Application.Init ();
                SetUpGui ();
15              Application.Run ();
            }
```

In den ersten drei Zeilen werden die benötigten Namespaces angegeben, vergleichbar mit den Packages unter Java. In Zeile sieben wird die Klasse ClientGui erzeugt, wo die einzelnen Parameter übergeben werden. Über die Funktion “Start” wird die Applikation initialisiert. Mit *Init* wird die Gtk# Bibliothek aufgerufen, mit *SetUpGui* aufgebaut und mit dem Aufruf *Run* gestartet. Der hier nicht aufgeführte Aufruf *Aplication.Quit* beendet die Ausführung.

```
            static void SetUpGui ()
2           {
                Gtk.Window w = new Gtk.Window ("Sign up");
4
                firstname_entry = new Entry ();
6                           ............

8                           VBox outerv = new VBox ();
                outerv.BorderWidth = 12;
```

```
10              outerv.Spacing = 12;
                w.Add (outerv);
12                          ............

14              VBox v = new VBox ();
                v.Spacing = 6;
16              h.PackStart (v, false, false, 0);

18              l = new Label ("_First name:");
                l.Xalign = 0;
20              v.PackStart (l, true, false, 0);
                l.MnemonicWidget = firstname_entry;
22                          ............

24              v = new VBox ();
                v.Spacing = 6;
26              h.PackStart (v, true, true, 0);

28              v.PackStart (firstname_entry, true, true, 0);

30                          ................
```

In der Methode SetUpGui wird die GUI zusammengesetzt. Es wird als Basis ein Gtk Window gebaut, in das vertikale Fenster (VBox) oder horizontale Fenster (HBOX) ineinander gebaut werden können, um eine sinnvolle Darstellung zu erhalten. Mit dem Aufruf *Packstart* werden einzelne Komponenten in die Fenster eingebunden. Der erste Parameter bindet das jeweilige Objekt ein. Die drei weiteren Parameter positionieren das Objekt im Fenster [Dumbill und Bornstein 2004]. Mit der Funktion *MnemonicWidget*, Zeile 21, werden den einzelnen Feldern Tastaturkürzel hinzugefügt, durch den Unterstrich vor “FirstName” in Zeile 18 wird der Buchstabe gekennzeichnet, der als Tastaturkürzel benutzt werden soll.

Im unteren Beispiel ist der Aufbau eines Event Handlers dargestellt. Die beiden Objekte `cancel` und `send` sind die beiden Button aus der Grafik 6.2. Es werden zwei neue EventHandler gebaut, die auf die Aktion *Clicked* die Funktionen aufrufen, die als Argument übergeben werden (*Cancle_Clicked, Send_Clicked*). Der *DeleteEvent* vom Hauptfenster (w) schließt die gesamte Anwendung über den X Button.

```
w.DeleteEvent += Window_Delete;
2 cancel.Clicked += new EventHandler (Cancel_Clicked);
  send.Clicked += new EventHandler (Send_Clicked);
```

Die aufgeführten Punkte sollen nur einen Einblick in die Funktionsweise geben und sind nicht vollständig.

Hotel

Auf dem Hotelserver werden verschiedene Komponenten von Mono benötigt. Im ersten Schritt werden die Assemblies für Webservices benötigt. Im Gegensatz zu dem mobilen Client muss hier zusätzlich zu dem Client der Server eingebunden werden. Dieser stellt die Angebote für den Gast im Hotelnetz zur Verfügung. Die Client Software, ”Client Proxy”, wird für die Kommunikation mit den Web-Services der Anbieter benutzt. Die Untersuchung des Ein- und Auslesens in, bzw. aus einer Datei, wird ebenfalls getestet. Es sollen die Gästedaten bei einer Reservierung dort zwischengespeichert und später zur Verifizierung ausgelesen werden.

In einem ersten Beispiel wird eine erste Funktion als Web-Service aufgebaut. Der Web-Service muß als ASMX Datei gekennzeichnet und im Hauptverzeichnis des Webervers abgelegt werden. Beim Aufruf des Dienstes wird ein String zurückgegeben.

```
1  <%@ WebService Language="C#" Class="Info" %>

3  using System;
   using System.Web.Services;
5
       [WebService ( Description = " erster Web-Services ")]
7      public class Info : WebService
       {
9
           [WebMethod (Description = "Das ist der erste")]
11         public string SampleMethod()
           {
13             return "Das ist ein Web-Service";
           }
```

In der ersten Zeile wird angegeben, dass es sich um einen Web-Service handelt und C# als Programmiersprache benutzt wird. Weiterhin wird der Name der Klasse angegeben. Im nächsten Schritt wird die Klasse aufgebaut, die als Webservice ausgewiesen wird. Die Methode *SampleMethod* gibt bei dem Aufruf ein einfachen String zurück.

Man kann den Web-Service über den Browser aufrufen und erhält folgende Ausgabe 6.3.

Man sieht auf der Seite eine Beschreibung des Webservices, die WSDL-Datei und kann sich automatisch über “Client Proxy” die Client Proxy Klasse für die Client Anwendung bauen lassen 6.4.

Der dabei generierte Code sieht wie unten aufgeführt aus. Man kann eine neue Klasse bauen, die ein Objekt aus dem Web-Service instanziert und die angebotenen Methoden aus dem Proxy aufruft. In diesem Beispiel wäre das die *SampleMethod*.

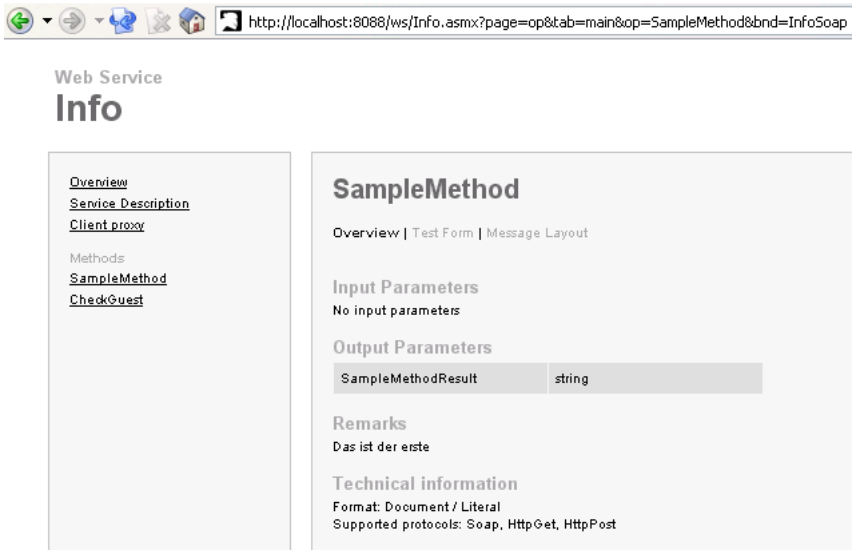

Abbildung 6.3.: Die Darstellung eines Web-Services im Browser

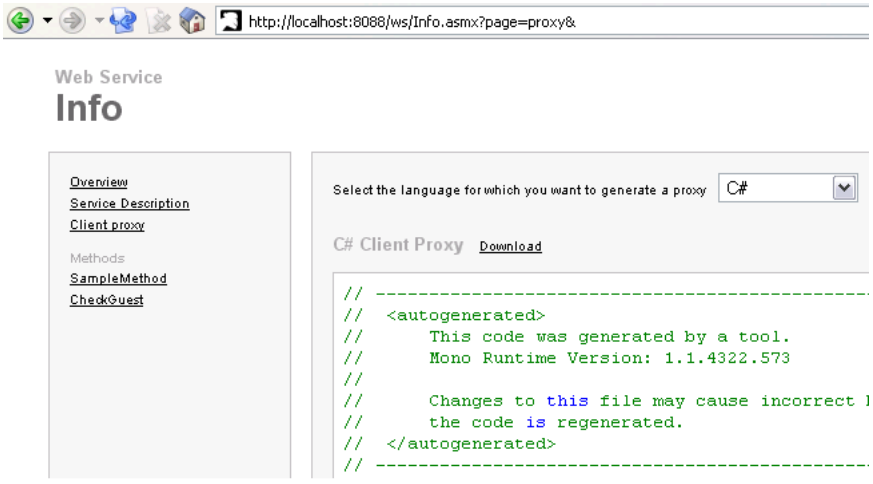

Abbildung 6.4.: Die Darstellung eines Proxy-Clients im Browser

Wichtig ist Zeile 10, in dieser Zeile wird angegeben, wo später der Service zu finden ist. Wenn diese Adresse nicht erreichbar ist, kann der Web-Service nicht benutzt werden.

```
//
------------------------------------------------------------------------------

2 //  <autogenerated>
  //      This code was generated by a tool.
4 //      Mono Runtime Version: 1.1.4322.573

6         ...............................
          ...............................
8 public class Info : System.Web.Services.Protocols.SoapHttpClientProtocol
    {

10      public Info () {
            this.Url = "http://wooster:8088/Info.asmx";
12      }

14          ...............................
            ...............................
16      public virtual string SampleMethod() {
            System.Object[] results = this.Invoke("SampleMethod", new object
                [0]);
18          return ((string)(results[0]));
        }
20
        public virtual System.IAsyncResult BeginSampleMethod(System.
            AsyncCallback callback, object asyncState) {
22          return this.BeginInvoke("SampleMethod", new object[0], callback,
                asyncState);
        }
24
        public virtual string EndSampleMethod(System.IAsyncResult asyncResult
            ) {
26          System.Object[] results = this.EndInvoke(asyncResult);
            return ((string)(results[0]));
28      }
```

Bei dieser Darstellung wurden die speziellen Konfigurationen für SOAP nicht aufgeführt, um eine bessere Übersicht zu erhalten. Der vollständige Code ist im Anhang einsehbar.

Für das Angebot an den Gast müssen weiterhin die aufgeführten Methoden aus dem Entwurf implementiert werden:

- ListOffers()
- BookABargain(Bargain reservation)
- SubmitTheReservation(Reservation reservation)
- GetStatusOfReservation(String reservationNumber)
- Cancel(String reservationNumber)

Bei der Methode **SubmitTheReservation(Reservation reservation)** müßen die Daten des Gastes mit der Reservierungsnummer zusammen abgespeichert werden. Später muss nachgewiesen werden, dass gebucht wurde, um die gebuchten Leistungen abzurechnen. Wie die Daten ausgelesen werden können, zeigt das nächste Beispiel. Es wird dabei nur der Vorgang des Einlesens beschrieben.

```
1  using System;
   using System.IO;
3  using System.Collections;

5  public class FileReadIn {
       public ArrayList ReadIn(String customFile) {
7          string file = customFile;
           ArrayList customList= new ArrayList();
9
           if (File.Exists(file)) {
11             using (TextReader reader = File.OpenText(file)) {
                   custom.Add(reader.ReadToEnd());
13                 reader.Close();
               }
15         } else {
               Console.WriteLine("keine Kundendatei zu finden");
17         }
       return customList;
19     }
   }
```

Es wird in der Methode *ReadIn* eine Pfadangabe übergeben, die in der Zeile zehn überprüft wird. Im nächsten Schritt wird über ein TextReader die angegebene Datei geöffnet und mit der Methode *ReadToEnd* Zeilenweise an die ArrayList übergeben. Dabei erkennt die Funktion jeweils das Ende der Zeile und arbeitet die Datei bis zu einem EOF ab.

Anbieter

Die Anbieter, in dieser Umgebung als Autovermietung realisiert, stellen ein Angebot für einen Gast zur Verfügung. Das Angebot wird dem Gast nicht direkt angeboten, sondern über den Ferienclub vermittelt. Die Autovermietung stellt dem Hotel die Angebote über den Web-Service `getSupplies(String hotel)` zur Verfügung. Beachtet werden muss bei der Implementierung des Web-Services, dass man nur einfache Datentypen nutzen kann. Die Daten müßen somit als primitiver Typ (String Array) übertragen werden. Die weitere Funktionalität ist die Abfrage der Buchung eines Autos durch das Hotel über die Web-Services `ValidateABargain(Bargain bargain)`, `BookABargain(Bargain aReservation)`. Es wird nur die Kommunikation zwischen dem Ferienclub und der Autovermietung über diese Web-Service abgebildet und keine weitere Anwendungslogik implementiert. Der Service `ValidateABargain(Bargain bargain)` gibt in dem Funktionsaufbau ein zufälliges ja oder nein für die Bestätigung oder Absage zurück. Mit dem Aufruf von `BookABargain(Bargain aReservation)` wird eine Zufallszahl, erzeugt die dem Hotel als Reservierungsnummer übergeben wird (primitiver Integer Typ, unten ein kurzes Beispiel).

```
[WebMethod (Description = "Die Reservierungsnummer fuer die Buchung
2             wird generiert und uebergeben")]
    public int BookABargain(Bargain aReservation)
4   {
        int reservationNumber = GenerateBargainNumber(aReservation);
6       return reservationNumber;
    }
```

6.2. Zusammenfassung

In diesem Prototypen wurde versucht, die Architektur 6.1 mit Hilfe von Mono aufzubauen.
Es war dabei nicht möglich eine Mono basierte Anwendung auf dem PDA zu implementieren, weil verschiedene Bibilotheken nicht verfügbar war. Alternativ wurde auf einen Linux basierten Laptop ausgewichen, der über eine spontanes Funknetz (Bluetooth) verfügte. Mit Hilfe des Frameworkes von Mono und der Programmiersprache C# sind die beschrieben Web-Services implementiert worden. Des Weiteren wurde eine GUI programmiert, die auf den GTK Bibilotheken beruht. Die gesamte Anwendung basiert auf Mono und ist unter Linux ebenso lauffähig wie unter Windows.

Es wurde mit Hilfe von Web-Services ermöglicht verschiedene Subsysteme miteinander zu verknüpfen, um ein Verteiltes System aufzubauen, das es wiederum ermöglichte diese direkt

miteinander interagieren zu lassen.
Mono hat als Plattform auf den verschiedenen Komponenten gedient.

Die Beurteilung von Mono als Plattform findet auf Basis der Erkenntnisse des Versuchsaufbaus statt und soll zeigen, ob die Kritikpunkte die positiven Aspekte überwiegen und somit Mono für heterogene Systemlandschaften dienen kann. Hierbei wäre dann denkbar, eine Plattform zu schaffen, die nicht windowsbasierten Systeme mit Microsofts .NET verbindet.

Fazit

Ziel dieser Arbeit war es zu untersuchen, ob Mono als Plattform für Webservices auf mobilen Endgeräten sinnvoll einsetzbar ist. Die Antwort darauf ist ein klares “Jein”. Es konnte leider keine Applikation, wie sie im Entwurf beschrieben oder auch im [Hildingson 2003] erläutert wurde, realisiert werden. Mono konnte auf dem benutzten PDA nicht eingesetzt werden, da die Ausführung verschiedener Bibliotheken nicht möglich war. Es handelte sich hierbei um die Bibliotheken für die GUI Anwendung und weiterhin um die Namespaces für die Web-Services. Leider kam trotz vieler neuer Monoversionen für den Personal Computer und den PowerPC, während dieser Arbeit keine neue Version vom PDA Mono heraus. Es wird zwar an diesem Paket gearbeitet, aber bis zu dem jetzigen Zeitpunkt wurde keine neue Version veröffentlicht.

Die Entwicklung von Anwendungen mit Mono und C# ist bei den angestellten Tests vollständig kompatibel zu der .NET Plattform. Es konnten Testprogramme jederzeit mit dem Compiler von Mono oder dem von .NET kompiliert werden. Die Microsoft Bibliotheken von .NET sind in Mono enthalten und nur durch Linux oder OpenSource spezifische Projekte, wie z. B. Mozilla Bibliotheken erweitert worden. Die Darstellung hierzu aus dem Kapitel 5.2 ist vollständig und wird ständig erweitert. Mono wurde in der Untersuchung unter verschiedenen Plattformen, wie Linux und Windows XP eingesetzt. Was weiterhin bei der Entwicklung auffiel war, dass viele der Dokumentationen nicht von Mono angeboten wurden, sondern vielmehr auf die von Microsoft angebotenen Hilfestellungen verwiesen. Zusätzlich zur Mono Referenz konnte oftmals die wesentlich umfangreichere .NET Dokumentation benutzt werden.

Bei der Entwicklung von graphischen Benutzeroberflächen bietet Mono mit GTK eine gut zu bedienende und gut dokumentierte Bibliothek an, die auf Grund der weitverbreiteten X-Oberfläche Gnome unter Linux zusätzlich ein breites Forum an Hilfen und Erweiterungen bietet. Die GTK-Bibliotheken enthält alle Funktionalität, die man für eine heutige GUI benötigt. Es wurden alle Funktionalitäten, wie man sie aus der Gnome Umgebung kennt implementiert und mit Glade ein Entwicklungswerkzeug mitgeliefert, das den Bau einer Anwendung gut unterstützt.
Es ist aber ebenfalls zu erwähnen, dass es unter Windows zu Schwierigkeiten kommen kann, wenn die zu benutzenden Bibliotheken nicht verfügbar sind oder eine nicht aktuelle Version

benutzt wird. Das Framework zum Bau von GUIs, wie diese unter .NET zur Entwicklung benutzt werden, ist im jetzigen Status von Mono leider nicht möglich. Mono wird in der nächsten offiziellen Version 1.2 diese unterstützen, wobei dieses, nur aus Kompatibilitätsgründen getan wird.

Die Web-Service wurden mit Hilfe von Active Server Pages (ASP) entwickelt und sind bei .NET über den Microsoft Webserver IIS nutztbar. Durch das Monomodul für den Apache Webserver ist es möglich gewesen, diese auf einen Apache Webserver auszuführen. Damit ist es zusätzlich möglich neben Web-Services auch Anwendungen mit ASP unter Linux zu entwickeln.
Die Werkzeuge, die Mono zur Entwicklung, z. B. von WSDL und Client Proxy bereitstellt sind ohne Schwierigkeiten einsetzbar und bieten viele Möglichkeiten der Erzeugung (entsprechend den Standards von W3C [WSDL:2001 2001] und [Soap:2003 2003]). Es war möglich zwischen den verschiedenen Rechnern über die Web-Services zu kommunizieren, unabhängig davon um welche Art Betriebssystem es sich handelt. Ebenso war es möglich die Web-Services über die beiden verschiedenen Webserver mod_mono für den Apache und XSP den Webserver von Mono anzubieten. Die verschiedenen Entwicklungsumgebungen unterstützten die Erzeugung weiterhin mit verschiedenen Erweiterung.

Die im Entwurf vorgestellte konzeptionelle Lösung für die Kommunikation von einem mobilen Endgerät über einen Server zu einem dritten Server mit Hilfe von Web-Services war in der Testumgebung prinzipiell möglich. Einschränkungen mussten wie schon beschrieben bei dem PDA gemacht werden. Die im Testszenarium beschrieben und in den Anforderungen wiedergegebenen Vorgaben konnten mit Mono realisiert werden. Die einzelnen Komponenten ließen sich ohne Schwierigkeiten implementieren und Mono bietet unter anderen Betriebssystem als Windows eine Möglichkeit der Sprachunabhängigkeit. Als weiterer Vorteil bietet sich dadurch die Möglichkeit, einzelne Windowsprogramme auf anderen Betriebssystemen zu nutzen.

Die Untersuchung hat gezeigt, dass es möglich ist mit Mono das in dieser Arbeit beschriebenen Szenario umzusetzen. Es gibt allerdings verschiedene Aspekte und Probleme, die hierbei nicht weiter berücksichtigt wurden. An erster Stelle muss dabei das Problem benannt werden, dass Mono auf dem PDA nicht lauffähig ist. Eine mögliche Lösung wäre vielleicht, die Anwendung auf einem Windows CE PDA unter .NET zu benutzen.

Eine weitere Fragestellung in diesem Zusammenhang ist die Geschwindigkeit mit der die Anwendungen laufen. Die Geschwindigkeit des Interpreters *mint* sowie die des Webservers muss untersucht werden. Weiterhin stellt sich die Frage, wie gut der Compiler *mcs* optimiert ist und wieviele Verbesserungen dort noch zu erwarten sind. Ebenfalls fehlt ein direkter Vergleich zu Microsofts .NET Plattform. Die Untersuchung hat zwar gezeigt, dass .NET Programme auf Mono lauffähig sind, aber nicht, wie schnell und wie stabil. Ein in dem Zusammenhang offengebliebener Test ist die Untersuchung der Skalierbarkeit. Bei den dynamisch

erzeugten Webseiten mit Hilfe von ASP ist zu vermuten, dass dieses über den Apache gut zu realisieren ist, aber eine gültige Aussage kann an dieser Stelle nicht getätigt werden.
Bei der Umsetzung des Entwurfes wurden verschiedene Aspekte nicht weiter untersucht, die aber für eine Anwendung in den heutigen Anwendungen eine gewisse Rolle spielen: Sessionmanagment, Sicherheit und die Möglichkeit der Interaktion mit verschiedenen Plattformen, wie z. B. Java. In diesem Bereich werden die WebServices eine wichtige Rolle spielen, da es z. B. in diesem hier benutzten Szenario ebenfall möglich wäre, dass ein Anbieter von Diensten Java als Technologie einsetzten könnte. Über die Web-Services bieten sich verschiedenste Möglichkeiten, alle Arten von Geräten und Plattformen miteinander kommunizieren zu lassen. Der hier vorgestellte Entwurf ist nur eine Möglichkeit.

7. Fazit

> “Jedes Ding hat zwei Seiten. Fanatiker sehen nur die eine.”
>
> Hellmut Walters

In dieser Arbeit wurde gezeigt, dass es durchaus möglich ist, den in der Einleitung angesprochenen Punkten, wie Internationalisierung, Verteilte Systeme und sich wandelnde Geschäftsprozesse mit Mono gerecht zu werden. Mit dem vorgestellten Szenario wurde versucht, die genannten Punkte abzubilden und eine möglichst realistische Testumgebung zu schaffen, um eine aussagekräftige Umgebung zur Evaluierung von Mono zu besitzen. Der vorgestellte Entwurf versuchte hierbei möglichst genau das geschilderte Szenario abzubilden. Die gewonnenen Erfahrungen bei der Umsetzung wurden im Kapitel 6.2 beschrieben.
Die Plattform selbst ist flexibel und bietet durch die Kompatibilität mit Microsofts .NET viele Möglichkeiten für Entwickler damit zu arbeiten. Die Entwicklung von Mono schreitet relativ schnell voran und hat mit Novell eine starke Unterstützung bei der Weiterentwicklung. Um einen besseren Eindruck von Mono zu erhalten, kann man sich bei Novell mehrere Projekte ansehen: iFolder, Beagle oder F-Spot [de Icaza 2005], die als Referenz dienen können was die Entwicklung mit Mono betrifft. Ein Vorteil den Mono gegenüber Microsoft besitzt ist, dass es ein OpenSource Projekt ist und dadurch auf eine breite Unterstützung von freien Entwicklern zurückgreifen kann und ein breites Forum existiert, um Probleme zu lösen oder aber um Erweiterungen, wie es z. B. von Java oder php her bekannt ist, anzubieten.

Im Hinblick auf die Zukunft existieren allerdings viele offene Fragen und es besteht Diskussionsbedarf. Es stellt sich die Frage, ob ein .NET unter Linux gewünscht ist. Mit .NET ist eine Basis geschaffen worden, auf der verschiedene Gedankenspiele zu Lizenzen und Digital Rights Management möglich sind. .NET ist schwerpunktmässig auf Web-Services aufgebaut und ausgelegt, um über das Internet zu agieren. Microsoft könnte damit alles modularisieren und den Zugriff auf Neuerungen im Bereich Software, vorausgesetzt ein Internetzugang ist vorhanden, über ihre eigenen Server vorgeben. Damit wäre gewährleistet, dass ein Einbinden von fremder Software auf Ihrem System nicht mehr möglich ist, ohne dass es vorher nicht von Microsoft “freigeben wurde”, da alles vor der Weiterverbreitung lizensiert werden müsste. Dieses würde im schlimmsten Fall bedeuten, dass auf windowsbasierten Systemen nichts mehr ohne die Zustimmung von Microsoft erfolgt. Ähnliche Möglichkeiten bieten

sich bei Benutzung verschiedener Angebote, die in anderen Bereichen miteingebunden sind. Ein Beispiel dafür wäre z. B. die Rechtschreibprüfung bei Word. Bei jeder Überprüfung der Rechtschreibung eines Dokumentes wird der Rechtschreibprüfer über Web-Service von den Microsoftservern angeboten und kostet eine bestimmte Gebühr. Wenn diese Dienste und Aufrufe in dem System mit integriert sind, muss der Benutzer entweder darauf verzichten oder die Gebühr entrichten [Fischer 2002].
Neben diesen hier genannten Kritikpunkten könnte man zweifeln, dass Microsoft mit .NET nur einen weiteren Marketing-Begriff für ihre Produkte eingeführt hat. Ausserdem stellt sich die Frage, ob Mono nicht Microsoft zuarbeitet, da es diese Plattform auf verschiedenen Systemen ausserhalb von Windows anbietet und etabliert. Bei einigen Bibilotheken und Modulen (z. B. Webservice und Sicherheit) erweitert und verbessert Mono zudem die bestehende Plattform. Es baut zusätzlich eine Community für .NET und C# auf. Das ist zum Vorteil von Microsoft und dient der weiteren Verbreitung ihrer Technologie. Mono erhält jedoch dafür von Microsoft keine Gegenleistung oder Unterstützung. Microsoft könnte dann z. B. an beliebigen Stellen sein System so ändern, dass Mono nicht mehr einsetzbar ist und viele Entwickler gezwungen werden sich weiter an Microsoft zu binden, statt eine freie Lösung zu benutzen (ein Beispiel dafür wäre die graphische Benutzeroberfläche).

Wagt man ein Blick auf die Zukunft für Mono, ergeben sich viele interessante Aspekte - im positiven wie im negativen Sinne. Die hier besprochenen Punkte sollen nur deutlich machen, dass es bei Mono nicht nur um eine Technologie, sondern auch um die Diskussion bezüglich freier Software geht. Interressant wird in der Zukunft die Frage sein, wie Microsoft sich selbst zu Mono stellt. Wird das Unternehmen Mono tolerieren, unterstützen oder versuchen es zu verbieten? Steve Ballmer äusserte sich eher negativ gegenüber Mono [Ballmer 2002]. Sollte Microsoft ihre Patente ändern, wäre Mono nicht abgeschrieben, sondern müsste einzelne Bestandteile umschreiben. Das würde im wesentlichen bedeuten, dass es nicht mehr dem Microsoftstandard entspricht, der aber unter Linux nicht zwingend benötigt wird.

Mono unterstützt zum jetzigen Zeitpunkt die Sprachunabhängigkeit und ist für verschiedene Plattformen erhältlich. Die Idee von Miguel de Icaza, die Sprachunabhängigkeit auch in anderen Systemen umzusetzen, ist gelungen und hat sicherlich eine spannende Zukunft vor sich. Es bleibt abzuwarten, ob es möglich ist mit Hilfe von Mono eine ähnliche Plattform aufzubauen, wie es Sun mit Java gelungen ist.

A. Anhang

A.1. Komponenten des Netzwerkes

Die Grafik A.1 zeigt die benötigten Komponenten und deren dazugehörigen Dienste, vom PDA ausgehend, die im Netz benötigt werden.

A.2. Wie wird Bluetooth installiert

An dieser Stelle wird auf Details der Bluetooth Konfiguration eingegangen, die im Kapitel Laborumgebung nicht weiter besprochen worden sind.

Bei der Entscheidung für ein Netzwerk mit Hilfe von Pan oder DUN ist PAN in dieser Umgebung die bessere Variante, da es eine TCP/IP Verbindung direkt auf dem L2CAP Protokoll[1] aufbaut und nicht wie bei DUN RFCOMM [2] zusätzlich das PPP (Point TO Point Protocoll) benutzt.

Installiert werden die Bluetooth Pakete auf dem PDA wie folgt:

```
ipkg -iv bluez-ussp-modules-2.4.18-rmk3
ipkg -iv bluez-modules-2.4.18-rmk3
```

[1] L2CAP Das Logical Link Control Adaption Protocol (L2CAP) adaptiert die oberen Protokolle an das Basisband bzw. Link Manager. Dabei werden den höheren Protokollen verbindungsorientierte und verbindungslose Services zur Verfügung gestellt. Segmentation, Reassembly, Multiplexing und Zusammenfassung der Teilnehmer in Gruppen wird unterstützt. L2CAP führt auch Quality of Service für die Parameter (Service Type, Verzögerungsbandbreite für Pakete, Verzögerung der Bits vom Sender zur Funkstrecke, Bandbreite u.ä.) durch [Weißensteiner 2005].

[2] RFCOMM ist ein einfaches Transportprotokoll, das eine serielle Schnittstelle (RS 232) über das L2CAP Protokoll emuliert. Es unterstützt bis zu 60 Verbindungen gleichzeitig zwischen zwei Bluetooth Geräten.[Weißensteiner 2005]

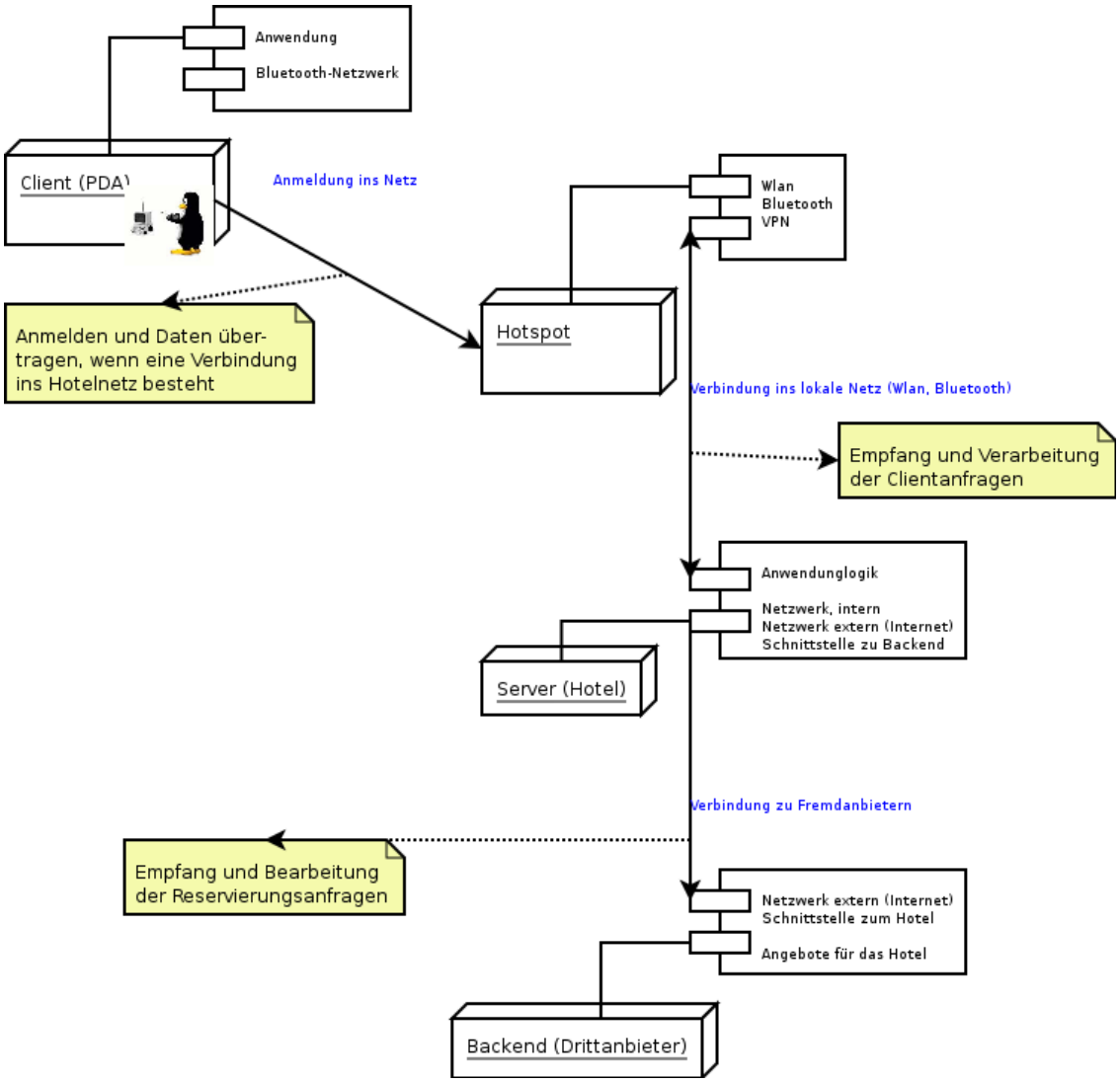

Abbildung A.1.: Komponenten für das Hotelnetz

Nach der Installation muss dem Hotelserver der sogenannte HCI-Daemon als Master konfiguriert werden. In dem hier dargestellten Ausschnitt der Konfigurationsdatei sind die grundsätzlichen Konfigurationen dargestellt.

```
# Default settings for HCI devices
device {
        # Local device name
        #   %d - device id
        #   %h - host name
        name "BA/AD ztnw335 (%d)";

        # Local device class
        class 0x100;

        # Inquiry and Page scan
        iscan enable; pscan enable;
        # Default link mode
        #   accept - always accept incoming connections
        #   master - become master on incoming connections
        lm accept,master;

        # Default link policy
        #   hold    - allow hold mode
        #   sniff   - allow sniff mode
        #   park    - allow park mode
        lp hold,sniff,park;


        # Authentication and Encryption
        auth enable;
        encrypt enable;
}
```

Auf der PDA-Seite werden die Grundeinstellung belassen und der PDA verbindet sich mit dem im Piconetz gefundenen Master.

Für die Annahme der Verbindung bzw. die Verifizierung benutzt der HCI-Daemon die sogenannte PIN Datei im /etc/bluetooth Verzeichnis [3].

[3] Es ist anzumerken, dass einige Geräte, wie mobile Telefone, nur numerische Pins aktzeptieren

Im nächsten Schritt muss auf beiden Seiten der Service Discovery Protocol Daemon (SDP Daemon)[4] gestartet werden. Über diesen Dienst finden die einzelnen Clients heraus welche Dienste mit welchen Eigenschaften verfügbar sind.

```
/etc/init.d/bluez-sdp start
```

Um jetzt Geräte in der Umgebung zu suchen benutzt man die Befehle *hcitool inq, hcitool scan* .Damit werden die Geräte, deren Namen und Ihre Adressen ermittelt.

```
# hcitool inq
Inquiring ...
00:02:C7:0B:86:AB       clock offset: 0x2c20    class: 0x120112

# hcitool scan
Scanning ...
        00:02:C7:0B:86:AB       familiar (0)

# l2ping 00:02:C7:0B:86:AB
Ping: 00:02:C7:0B:86:AB from 00:02:72:01:44:CF (data size 20) ...
20 bytes from 00:02:C7:0B:86:AB id 200 time 33.87ms
20 bytes from 00:02:C7:0B:86:AB id 201 time 38.12ms
20 bytes from 00:02:C7:0B:86:AB id 202 time 32.13ms
3 sent, 3 received, 0% loss
```

Um den Master zu aktivieren, wird jetzt auf dem Hotelserver der dritte Dienst gestartet, der Personal Area Network Daemon, kurz PAND.

```
# pand -s -r NAP -M
# ps axf | grep pand
14731 pts/3    S+     0:00          \_ grep pand
14729 ?        Ss     0:00 pand -s -r NAP -M
```

Mit diesem Befehl wartet der Master auf einkommende Verbindungen. Die “Schalter” haben dabei folgende Bedeutung:

- **-s**: warten auf einkommende Verbindung
- **- M**: ist ein Master

[4] Dient zur Lokalisierung von Diensten; von der Anwendungsschichten verwendet. Er setzt auf L2CAP auf.

- **-r NAP**: steht für eine genaue Angabe des Dienstes, in diesem Beispiel ein Network Access Point (NAP)

Will ein Client sich jetzt im Netzwerk anmelden, muss im ersten Schritt der Dienst gesucht werden.

```
# hcitool inq
Inquiring ...
00:02:72:01:44:CF       clock offset: 0x2c20    class: 0x120112
```

Danach verbindet man sich mit

```
pand -c 00:02:72:01:44:CF
```

Mit dem Befehl

```
# pand -l
bnep0 00:02:72:01:44:CF NAP
```

erfährt man, ob die Verbindung zustande gekommen ist. Bei einer erfolgreichen Verbindung erhält man mit bnep0 eine Netzwerkschnittstelle die, wie z. B. eth0 mit ifconfig, konfiguriert werden kann. Zur Vereinfachung dieser Konfiguration werden diese Schnittstellen über Hotplug konfiguriert.

Will man mit dem Client auf das gesamte Netzwerk zugreifen, muss auf dem Server IP Forwarding und Masquerading aktivieren werden:

```
# echo "1" > /proc/sys/net/ipv4/ip_forward
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

A.3. Entwicklungsumgebungen

SharpDevelop

Die Entwicklungsumgebung SharpDevelop für .NET und Mono unter Windows A.2.

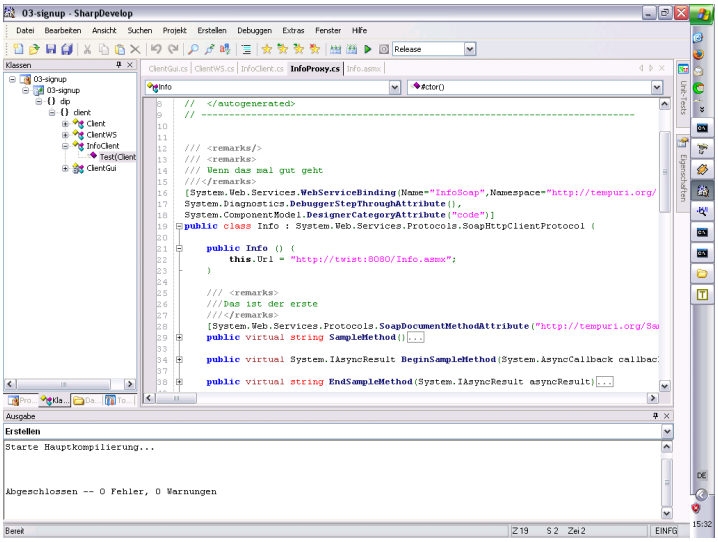

Abbildung A.2.: SharpDevelop unter Windows

Monodevelop

Die Entwicklungsumgebung Monodevelop für Mono unter Linux A.3.

Glade

Die Entwicklungsumgebung Glade für Mono und Gtk unter Linux und Microsoft A.4.

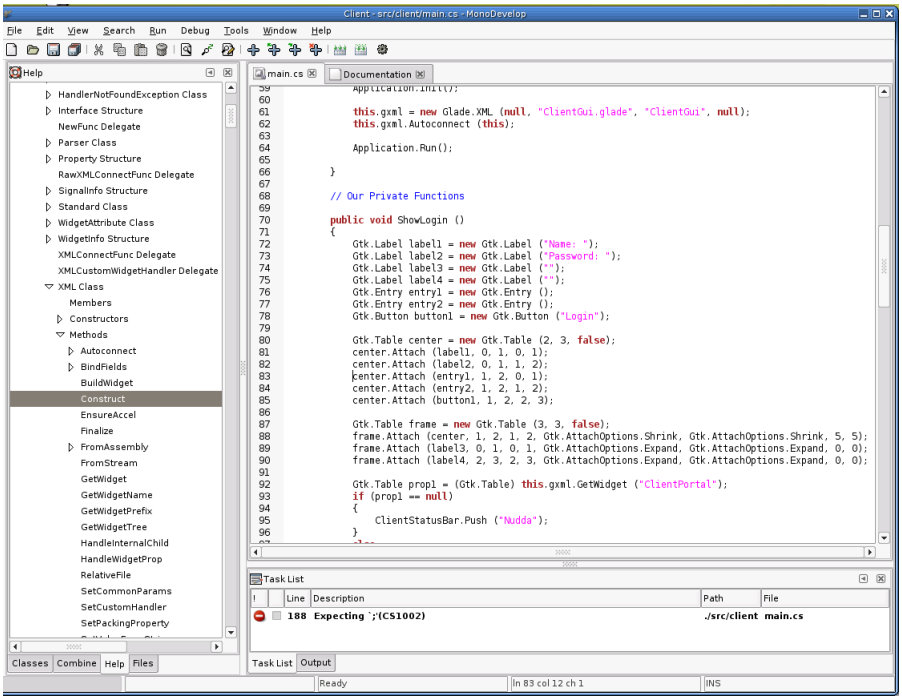

Abbildung A.3.: Monodevelop unter Linux

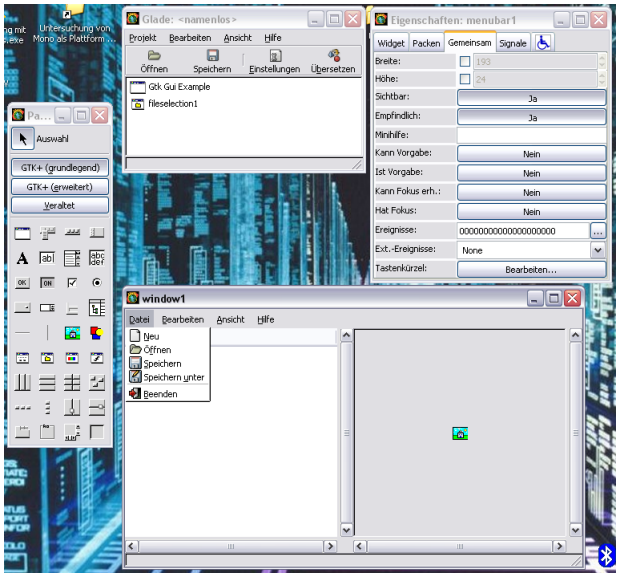

Abbildung A.4.: GUI mit Glade erstellt

A.4. Inhalt der CD-ROM

Im Wurzelverzeichnis der CD-ROM befinden sich die Verzeichnisse “Diplomarbeit”, “Implementierung” und “Quellen”.

Der Ordner Diplomarbeit enthält die Arbeit im PDF-Format. Im Verzeichnis Implementierung befinden sich die prototypische Implementierung aus dem Kapitel 6. Es sind jeweils die einzelnen Implementierung als Quellcode abgelegt sowie ein Installationsprogramm für Mono unter Windows und Linux [mono-project:2004 2004] beigelegt.
Das Verzeichnis Quellen beinhaltet einige der Dokumente, die im Literaturverzeichnis erwähnt werden.

Literaturverzeichnis

WSDL:2001 2001 : *Web Services Description Language (WSDL) 1.1*. 2001. – URL http://www.w3.org/TR/2001/NOTE-wsdl-20010315. – Zugriffsdatum: 2005-04-26

CIL:2002 2002 Standard ECMA-335, Common Language Infrastructure (CLI) / ecma-international. URL http://www.ecma-international.org/publications/standards/Ecma-335.htm. – Zugriffsdatum: 2005-04-23, 2002. – Forschungsbericht

Soap:2003 2003 : *SOAP Version 1.2 Part 1: Messaging Framework*. 2003. – URL http://www.w3.org/TR/soap12-part1/. – Zugriffsdatum: 2005-04-24

systinet:2003 2003 PAPER, A Systinet W. (Hrsg.): *Web Services: A Practical Introduction*. 2003. – URL http://www.systinet.com. – Zugriffsdatum: 2005-03-02

ocl:2004 2004 INTEL (Hrsg.): *C# Class Libary*. 2004. – URL http://ocl.sourceforge.net/. – Zugriffsdatum: 2004-10-14

Wiki.dos:2004 2004 WIKIPEDIA, DOS (Hrsg.): *Denial of Service, Wikipedia, der freien Enzyklopädie*. 2004. – URL http://de.wikipedia.org/wiki/DOS. – Zugriffsdatum: 2004-10-27

dotgnu:2004 2004 DOTGNU (Hrsg.): *DOTGnuProject*. 2004. – URL http://www.dotgnu.org. – Zugriffsdatum: 2004-10-14

Mono-book:2004 2004 MONO HANDBOOK (Hrsg.): *The Mono Handbook*. 2004. – URL http://www.gotmono.com/docs/. – Zugriffsdatum: 2004-09-27

mono-project:2004 2004 ABOUT MONO (Hrsg.): *The Mono Project*. 2004. – URL http://www.mono-project.com/about/index.html. – Zugriffsdatum: 2004-10-14

MonoHandhelds:2004 2004 MONO::HANDHELDS (Hrsg.): *mono::handhelds*. 2004. – URL http://amy.udd.htu.se/~malte/mono/. – Zugriffsdatum: 2004-09-27

Wiki.net:2004 2004 WIKIPEDIA, .NET (Hrsg.): *.NET*. 2004. – URL http://de.wikipedia.org/wiki/Dotnet. – Zugriffsdatum: 2004-10-27

skale:2004 2004 ABOUT .NET (Hrsg.): *Was ist .NET*. 2004. – URL http://msdn.microsoft.com/library/deu/default.asp?url=/library/DEU/vsent7/html/vxcondesigningforscalability.asp. – Zugriffsdatum: 2004-10-15

.net:2004 2004 ABOUT .NET (Hrsg.): *Was ist .NET*. 2004. – URL http://www.microsoft.com/germany/themen/net/wasistdotnet.mspx. – Zugriffsdatum: 2004-10-15

Schema:2004 2004 : *XML Schema Part 2: Datatypes Second Edition*. jun 2004. – URL http://www.w3.org/TR/xmlschema-2/. – Zugriffsdatum: 2005-04-24

Sig:2005 2005 BLUETOOTH SIG (Hrsg.): *Bluetooth SIG*. 2005. – URL http://german.bluetooth.com/about/members.htm#top. – Zugriffsdatum: 2005-04-23

Bbox:2005 2005 BUSYBOX (Hrsg.): *BusyBox*. 2005. – URL http://www.busybox.net/. – Zugriffsdatum: 2005-04-23

Gnome:2005 2005 GNOME (Hrsg.): *GNOME*. 2005. – URL www.gnome.org. – Zugriffsdatum: 2005-04-23

GPE:2005 2005 GPE (Hrsg.): *GPE, X11*. 2005. – URL http://gpe.handhelds.org/. – Zugriffsdatum: 2005-04-23

Handheld:2005 2005 HANDHELD (Hrsg.): *Handhelds.org*. 2005. – URL http://www.handhelds.org/geeklog/index.php. – Zugriffsdatum: 2005-04-23

Ipaq:2005 2005 IPAQ (Hrsg.): *ipaqlinux*. 2005. – URL http://www.ipaqlinux.com/. – Zugriffsdatum: 2005-04-23

Ipkg:2005 2005 IPGKFIND (Hrsg.): *ipkgfind*. 2005. – URL http://ipkgfind.handhelds.org/. – Zugriffsdatum: 2005-04-23

.netProg:2005 2005 .NET (Hrsg.): *.NET Framework Developer's Guide*. 2005. – URL http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconProgrammingWithNETFramework.asp. – Zugriffsdatum: 2005-03-23

OPIE:2005 2005 OPIE (Hrsg.): *OPIE, X11*. 2005. – URL http://opie.handhelds.org. – Zugriffsdatum: 2005-04-23

Kde:2005 2005 TROLLTECH (Hrsg.): *Trolltech*. 2005. – URL www.trolltech.com/. – Zugriffsdatum: 2005-04-23

Ballmer 2002 BALLMER, Steve: Wir machen Software. In: *iX* (2002)

Bresch 2004 BRESCH, Marco: *Erweiterung des JavaServer Faces Frameworks für den Einsatz in mobilen Anwendungen*, Hochschule für Angewandte Wissenschaften Hamburg, Dissertation, oct 2004. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/diplom/bresch.pdf. – Zugriffsdatum: 2005-01-16

Cabrera 2004a CABRERA, Luis F.: Web Services Atomic Transaction (WS-AtomicTransaction) / IBM developerWorks. nov 2004. – Forschungsbericht

Cabrera 2004b CABRERA, Luis F.: Web Services Business Activity Framework / IBM developerWorks. nov 2004. – Forschungsbericht

Czarski 2005 CZARSKI, Carsten: Webservices überbrücken technische Gräben. In: *Computer Zeitung* (2005)

Dumbill und Bornstein 2004 DUMBILL, Edd ; BORNSTEIN, Niel M.: *Mono*. 1. Auflage. O'Reilly, 2004 (A Developer's Notebook). – ISBN 0-596-00792-2

Ferrara und MacDonal 2003 FERRARA, Alex ; MACDONAL, Matthew: *Programmierung von .NET Web Services*. 1. Auflage. O'Reilly, 2003. – ISBN 3-89721-348-6

Fischer 2002 FISCHER, Thorsten ; ABOUT MONO (Hrsg.): *Mono und Gnome - der gegenwärtige Stand der Dinge/Scharfes Gtk+*. 2002. – URL http://www.linux-magazin.de/Artikel/ausgabe/2002/07/mono/mono.html?print=y. – Zugriffsdatum: 2005-04-08

Gamma Erich und Vlissides 1995 GAMMA ERICH, Ralph J. ; VLISSIDES, John: *Design Patterns: Elements of Reusable Object-Oriented Software*. 1. Auflage. Addison Wesley, 1995

George Coulouris 2002 GEORGE COULOURIS, Jean Dollimore und Tim K.: *Verteilte Systeme, Konzept und Design*. 3. Auflage. Addison-Wesley, 2002. – ISBN 3-8273-7022-1

Gettys 2005 GETTYS, Jim: Familiar v0.8.1 Installation / handhelds.org. URL http://www.handhelds.org. – Zugriffsdatum: 2005-03-02, 2005. – Forschungsbericht

Gustavo Alonso und Machiraju 2004 GUSTAVO ALONSO, Harumi K. ; MACHIRAJU, Vijay: *Web Services - Concepts, Architectures and Applications*. 1. Auflage. Springer-Verlag Berlin Heidelberg New York, 2004. – ISBN 3-540-44008-9

Hildingson 2003 HILDINGSON, Malte: *Bring the Monkey to the iPAQ*. August 2003. – URL http://www.sys-con.com/story/?storyid=38933&DE=1. – Zugriffsdatum: 2005-03-24

Hilzinger 2004 HILZINGER, Marcel: Blaue Netze. In: *Linux Magazin, network edition* (2004)

hotel.de 2003 HOTEL.DE: *hotel.de Newsletter*. Juli 2003. – URL http://media.hotel.de/Hotels-in/WLAN_Hotel.htm. – Zugriffsdatum: 2005-03-24

de Icaza 2005 ICAZA, Miguel de: Mono at ApacheCon / novell. jan 2005. – Forschungsbericht

Knuth 2003 KNUTH, Michael: *Web Services - Einführung und Übersicht*. 2. Auflage. Software und Support Verlag, 2003. – ISBN 3-935042-38-8

Kulicke 2005 KULICKE, Marcel: .Net Framework und die Programmiersprache C# / TU Berlin. 2005. – Forschungsbericht

Little und Freund 2003 LITTLE, Mark ; FREUND, Thomas J.: A comparative analysis of WS-C/WS-Tx and OASIS BTP / IBM developerWorks. oct 2003. – Forschungsbericht

Lüpke 2004 LÜPKE, Andre: *Entwurf einer Sicherheitsarchitektur für den Einsatz mobiler Endgeräte*, Hochschule für Angewandte Wissenschaften Hamburg, Dissertation, apr 2004. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/diplom/luepke.pdf. – Zugriffsdatum: 2004-08-09

Mählmann 2004 MÄHLMANN, Lars: Sichere Übertragung im WLan mit mobilen Endgeräten (speziell unter Linux). URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/studien/maehlmann.pdf. – Zugriffsdatum: 2004-08-09, jun 2004. – Forschungsbericht

Ogbuji 2003 OGBUJI, Uche: *The Past, Present and Future of Web Services*. 2003. – URL http://www.webservices.org. – Zugriffsdatum: 2005-03-02

Sherman R. Alpert und Woolf 1998 SHERMAN R. ALPERT, Kyle B. ; WOOLF, Bobby: *The Design Patters Smalltalk Companion*. 1. Auflage. Addison Wesley, 1998. – ISBN 0-201-18462-1

Snell 2002 SNELL, James: An introduction to BPELWS, WS-Coordination, and WS-Transaction / IBM developerWorks. aug 2002. – Forschungsbericht

Weiser 1991 WEISER, Mark: *The Computer for the 21st Century*. 1991. – URL http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html. – Zugriffsdatum: 2005-04-08

Weißensteiner 2005 WEISSENSTEINER, Hansi: Service Location Protocols GSD - Bluetooth SDP / Digital Enterprise Research Institute. jan 2005. – Forschungsbericht

Wikipedia 2005 WIKIPEDIA, der freien E.: *Monolithischer Kernel*. 2005

Wirdemann 2004 WIRDEMANN, Ralf: EJB-Dienste als Web Service / IX, Heise Zeitschriften Verlag GmbH & Co. KG. jun 2004. – Forschungsbericht

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 4. Juni 2004	
Ort, Datum	Unterschrift