universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Modellierung mit mobilen Devices: Ein Technologievergleich und eine Applikationsbewertung“

verfasst von / submitted by

Markus Köhler Bakk.rer.soc.oec.

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Diplom-Ingenieur (Dipl.-Ing.)

Wien, 2015 / Vienna, 2015

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet	A 066 926
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Master Wirtschaftsinformatik (UG2002)
Betreut von / Supervisor:	o. Univ.-Prof. Dr. Dimitris Karagiannis

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe.

Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Weiters wurde diese Arbeit keiner anderen Prüfungsbehörde übergeben und auch noch nicht veröffentlicht.

Wien, den 01.10.2015

Danksagung

Ich möchte mich ganz besonders bei meiner Mutter, meinem Vater, meiner Familie und meiner Freundin Cornelia für die Unterstützung, während des ganzen Studiums, bedanken. Bei meiner Mutter möchte ich mich auch für das mehrmalige Korrekturlesen von meinen diversen wissenschaftlichen Arbeiten bedanken.

Weiters gilt mein Dank meinem Betreuer o. Univ.-Prof. Dr. Dimitris Karagiannis, der mich während meines Studiums immer unterstützt hat. Bedanken möchte ich mich auch für die Möglichkeiten mit Ihm am Institut für Knowledge Engineering zu arbeiten. Vielen Dank auch an Hanno für die Hilfe und den vielen Input zu dieser Arbeit.

Inhaltsverzeichnis

Abkürzungsverzeichnis

App Applikation

Apps Applikationen

BPEL Business Process Execution Language

BPMN Business Process Model and Notation

DTD Dokumenttypdefinition

E-Mail Electronic Mail

EDGE Enhanced Data Rates for GSM Evolution

FDM Frequenzmultiplexing

Full-HD Full High Definition

GPRS General Packet Radio Service

GSM Global System for Mobile Communications

HP Hewlett Packard

HSPA High Speed Packet Access

HSUPA High Speed Uplink Packet Access

HTTP Hypertext Transfer Protocol

J2ME Java Platform 2, Micro Edition

JSON JavaScript Object Notation

kBit/s Kilobit pro Sekunde

LTE Long Term Evolution

MBit/s Megabit pro Sekunde

MIMO Multiple Input & Multiple Output

mm Millimeter

MOF Meta Object Facility

Mrd. Milliarden

MS Microsoft

OFDM Orthogonales Frequenzmultiplexverfahren

OMG Object Management Group

PDUs Protokolldateneinheiten

POJO Jsonschema2pojo

ppi	pixel per inch
RDF	Resource Description Framework
REST	Representational State Transfer
RSS	Really Simple Syndication
SMS	Short Message Service
SOAP	Simple Objekt Access Protocol
SVG	Scalable Vector Graphics
TDM	Zeitmultiplexing
TNC	Transnational Corporations
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
UMTS	Universal Mobile Telecommunications System
URI	Uniform Resource Identifier
WAP	Wireless Application Protocol
WWW	World Wide Web
XAML	Extensible Application Markup Language
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language

1. Motivation und Problemstellung

In der heutigen Zeit sind Mobiltelefone aus dem Alltag nicht mehr wegzudenken. Ihre Technologie hat sich in den letzten Jahren sehr stark weiterentwickelt und immer mehr an Funktionalität dazu gewonnen. Auch die Hardware der Telefone wurde immer besser und vor allem die Displays der Endgeräte sind mit den Anfängen nicht mehr vergleichbar. Im Gegensatz zu früheren Jahren gibt es bereits mobile Endgeräte, die „nur noch" aus einem Display bestehen. Auf diesen Geräten kann man nicht nur die Inhalte immer besser erkennen, sondern man kann mit ihnen auch arbeiten. Mobiltelefone mit immer mehr Funktionen werden seit einigen Jahren Smartphones genannt. Das sogenannte Smartphone wird im (Gabler Lexikon Medienwirtschaft) definiert als „Mobiltelefon mit erweitertem Funktionsumfang. Dazu zählen neben der Telefonie und Short Message Service (SMS) üblicherweise Zusatzdienste wie Electronic Mail (E-Mail), World Wide Web (WWW), Terminkalender, Navigation sowie Aufnahme und Wiedergabe audiovisueller Inhalte." (Zitat 2. Absatz „Ausführliche Erklärung"(Wirtschaftslexikon Online Smartphone)) Weiters erwähnen die Autoren, dass das Betriebssystem[1] auf diesen mobilen Geräten immer aufwendiger und komplexer wird. Die Betriebssysteme sind zum Beispiel Symbian OS von Nokia, Blackberry OS von RIM, iPhone OS von Apple oder Android von Google. Durch das Angebot von diversen Applikationen (Apps) bieten die Smartphones dem Benutzer sehr viele unterschiedliche Einsatzmöglichkeiten. Diese Applikationen sind sowohl für berufliche als auch für private Zwecke in der Freizeit einsetzbar.

Heutzutage steigen die Verkaufszahlen der Smartphones jährlich (Heise.de Handy-Verkaufszahlen) und jene der „normalen" Mobiltelefone, umgangssprachlich auch Handy genannt, fallen. Smartphones werden immer mehr eingesetzt, sowohl in Firmen als auch im privaten Bereich. Die Verkaufszahlen sprechen dafür, dass man dieses Marktsegment auf keinen Fall unterschätzen und nicht außer Acht lassen darf. (siehe auch Kapitel 2.5 Die Wirtschaftliche Situation von Applikationen) Vor allem den Markt für die bereits erwähnten Applikationen sollte man nicht unterschätzen, denn laut dem Marktforscher Gartner bietet er sehr viel Potential. Allerdings veranschaulicht Abbildung 1 auch sehr

[1] Operating System (deutsch: Betriebssystem): Ein Betriebssystem ist notwendig für die Benutzung eines Computers. Auch die neuen Smartphones besitzen ein Betriebssystem, das Arbeitsspeicher, Prozessor und die restlichen Hardwarekomponenten verwaltet. Das OS verbindet die Hardwarekomponenten mit der Anwendungssoftware des Benutzers. (Wirschaftsinformatik I)

deutlich, dass die Hersteller und die unterschiedlichen Betriebssysteme verschiedene Entwicklungen durchgemacht haben. (siehe auch Kapitel 2.5 Die Wirtschaftliche Situation von Applikationen)

Das Arbeiten auf mobilen Endgeräten wird in jeder Branche immer attraktiver. In den Zeiten von „Transnational Corporations“ (TNC) sind Dienstreisen keine Seltenheit geworden. Um während dieser Reise auch produktiv arbeiten zu können, benötigt man dementsprechende Geräte, Software oder Applikationen. Mit Hilfe von Mobiltelefonen können Arbeitnehmer immer und überall erreicht werden. Es besteht nicht nur die Möglichkeit Probleme zu besprechen, sondern deren Lösung unmittelbar umzusetzen, wie zum Beispiel ein fehlerhaftes Datenmodell oder einen Prozess zur Fehlerbehebung zu koordinieren. Derzeit gibt es im beruflichen Alltag jedoch keine Smartphone-Applikation, die man als „optimal“ oder gut einsetzbar für Modelle[2] einstufen könnte. Momentan wird eine Handvoll Modellierungsapplikationen für Smartphones angeboten, jedoch fehlen diesen Schnittstellen, abhängig von der Modellgröße, Übersichtlichkeit, Zugriff und Ausfallssicherheit beim Bearbeiten von bereits auf einem PC erstellten Modellen. Durch die schnelle Bewegung zwischen den unterschiedlichen mobilen Internetgeschwindigkeiten muss ein Management der Datenpakete und vor allem der Größe dieser Datenpakete eingerichtet werden. Aktuell gibt es noch keine freiverfügbare Applikation (App), die alle diese Anforderungen erfüllen kann. Im Rahmen dieser Arbeit hat sich der Autor daher mit der Entwicklung und der Umsetzung einer solchen Applikation auseinander gesetzt. In diesem Zusammenhang gilt es sich vier essentielle Fragen zu stellen:

1. Wie kann man eine Modell-Notation für kleine Devices anpassen, was ist zu berücksichtigen?
2. Welche Kommunikationsprotokolle und Kommunikationstechnologien eignen sich am besten, um den Anforderungen zu entsprechen?
3. Wie hat die Modell-Datenstruktur für mobiles Environment auszusehen und welche spezielle Anforderungen müssen erfüllt werden?
4. Wie können die Technologien und Protokolle verglichen werden und in Zusammenhang mit Applikationen bewertet werden?

[2] Der Begriff Modell wird in Kapitel 2.1 näher beschrieben.

Die Motivation für diese Arbeit ist das Bestreben, das mobile Arbeiten, dessen Bedeutung in der heutigen Zeit stetig wächst, bestmöglich zu unterstützen. Das Ziel ist daher aufzuzeigen, wie wichtig Modellieren auf mobilen Devices ist und wie man dies möglichst benutzerfreundlich umsetzen kann.

2. Einleitung

In der Einleitung werden die wichtigsten Begriffe und Hintergründe der Arbeit kurz beschrieben. Es soll eine Einführung in die Begriffe der Open Source und Open Models geben. Weiters wird der Nutzen eines Modells herausgearbeitet und definiert, was es mit dem Begriff Metamodell auf sich hat. Dies legt den Grundstein, um in der folgenden Arbeit Modellnotation auf kleinen Devices zu diskutieren und eventuelles Verbesserungspotential aufzuzeigen. Da Bildschirme relative kleine Displays haben, muss man der Frage nachgehen, ob die aktuellen Modellnotationen nicht auf kleine Devices verbesserungswürdig sind. Damit man aber auch auf kleinen Devices mit Modellen arbeiten kann, muss man sich mit der Beladung von Daten auseinandersetzen, wie kann dies passieren und welcher Weg ist der schnellste und effektivste. Natürlich muss die Beladung der Daten über drahtlose Netzwerke möglich sein, da man diese Daten mobil nutzen möchte. Um erhaltene Daten verarbeiten zu können, müssen Anforderungen an eine Modelldatenstruktur festgelegt werden. Im Abschluss der Arbeit werden die erwähnten Methoden und Technologien verglichen und bewertet. Weiters soll ein Überblick gegeben werden, welche Anforderungen eine Modellierungsapplikation erfüllen muss.

2.1. Was ist ein Modell und welchen Nutzen hat es?

Das Wort Modell kommt ursprünglich vom lateinischen Ausdruck „modulus“ (wörtlich übersetzt: das *Maß*). Ein Modell ist eine Abbildung der Wirklichkeit. (Skriptum Modellierung) (Skriptum Unternehmensmodellierung) Nach dem deutschen Philosophen Herbert Stachowiak wird ein Modell durch 3 wesentliche Merkmale gekennzeichnet:

1. Abbildungsmerkmal
   „Modelle sind stets Modelle von etwas, nämlich Abbildungen, Repräsentationen natürlicher oder künstlicher Originale, die selbst wieder Modelle sein können.“ (Zitat (Allgemeine Modelltheorie) S131 2. Absatz) Der Autor spricht hier von Originalen, die entweder in der Natur vorkommen können oder aber auch technisch vom Menschen geschaffen sind. Sie können sowohl physisch existieren, eine gedankliche Idee oder auch in der Fantasy entsprungen sein. Ein Modell hat also den Zweck, etwas abzubilden.

2. Verkürzungsmerkmal

   „Ein Modell erfasst im Allgemeinen nicht alle Attribute des durch sie repräsentierten Originals, sondern nur solche, die den jeweiligen Modellschaffern und/oder Modellbenutzern relevant scheinen." (Zitat (Allgemeine Modelltheorie) S132 2. Absatz) Der Autor möchte hier zum Ausdruck bringen, dass ein Modell nie alle Attribute des Originals übernimmt, sondern nur solche, die dem Ersteller wichtig erscheinen. Zur Nachvollziehbarkeit dieser Entscheidung muss man anmerken, dass nur derjenige dies verstehen kann, der Original und Modell kennt. Die Verkürzung ist ein wesentliches Merkmal des Modells, da es nur ein Abbild der Wirklichkeit ist und keine Kopie des Originals darstellen soll. Bei den Attributen des Originals muss es sich auch nicht unbedingt um dieselben des Modells handeln, da der Modellschaffer Attribute unter gewissen Punkten zusammenfassen kann, um dem Modell mehr Aussagekraft zu geben. Daher muss nicht jedes Attribut des Modells zwangsweise auch im Original vorkommen.

3. Pragmatisches Merkmal

   „Modelle sind ihren Originalen nicht per se eindeutig zugeordnet. Sie erfüllen ihre Ersetzungsfunktion a) für bestimmte - erkennende und/oder handelnde, modellbenutzende - Subjekte, b) innerhalb bestimmter Zeitintervalle und c) unter Einschränkung auf bestimmte gedankliche oder tatsächliche Operationen." (Zitat (Allgemeine Modelltheorie) S132-S133 3. Absatz -1.Absatz) Der Autor möchte mit dem Punkt a) zum Ausdruck bringen, dass ein Modell nicht nur ein Modell von etwas ist, sondern auch ein Modell für jemanden darstellt. Der Punkt b) spiegelt die Zeit oder das Zeitintervall wider, für die das Modell benötigt wird. Der dritte Punkt c) soll aufzeigen, dass ein Modell einen bestimmten Zweck erfüllen soll. Kurz kann man zusammenfassen, dass der Punkt a) für wen? , Punkt b) wann? und der Punkt c) wozu? als Fragewort darstellt. (Allgemeine Modelltheorie)

Diese drei Merkmale gelten für jede Art von Modellen und sind nicht speziell für die Branche der Informatik entwickelt worden. Man kann jedoch in allen drei Merkmalen eine eindeutige Notwendigkeit für Modelle in der Informatik erkennen.

Beinahe in jeder Wissenschaft kommen Modelle zur Anwendung und helfen dabei komplexe Szenarien zu veranschaulichen und zu verstehen. Angefangen bei einfachen

mathematischen Problemen bis hin zu komplexen Atom-Modellen in der Physik oder einem Kugel-Modell eines Gases in der Chemie. Modelle erleichtern den wissenschaftlichen Alltag maßgeblich und sind in der heutigen Zeit nicht mehr wegzudenken. Verschiedene Philosophen haben bereits die Wichtigkeit von Modellen festgestellt und auch in dieser Wissenschaft wurden bereits etliche Modelle entwickelt. In der Philosophie hat man sich auch mit dem Thema der Modellierung und mit den Regeln der Modellerstellung befasst.

In der Quelle (Stanford Models Science) wird in Kapitel 2.3 auf die Strukturierung eines Modells eingegangen. Bei der Strukturierung eines Modells und bei Regeln zur Modellierung kommt man in der Informatik automatisch auf das Thema Metamodell.

2.2. Was ist ein Metamodell und welchen Nutzen hat es?

Das Wort „meta“ kommt ursprünglich aus dem Griechischen und wird im Deutschen als Vorsilbe mit unterschiedlichen Bedeutungen verwendet. Es kann „inmitten“ (räumlich), „zugleich mit“, „auf (etwas) los“ oder „nach“ (zeitlich oder auch in einer Rangfolge)bedeuten. Im Fall des Metamodells ist die Übersetzung „nach“ (Rangfolge) am treffendsten. Dieses „nach“ oder auch „hinter“ beschreibt auch den entwickelten Standard Meta Object Facility (MOF) der Object Management Group (OMG) am besten. Die MOF umschreibt ein eigenes für Metadaten entwickeltes Architekturgerüst mit dem man Metamodelle und Meta^{2}modelle entwickeln kann. (OMG Metamodell) Im MOF Standard geht man von vier Ebenen, wie in Abbildung 1 dargestellt, aus.

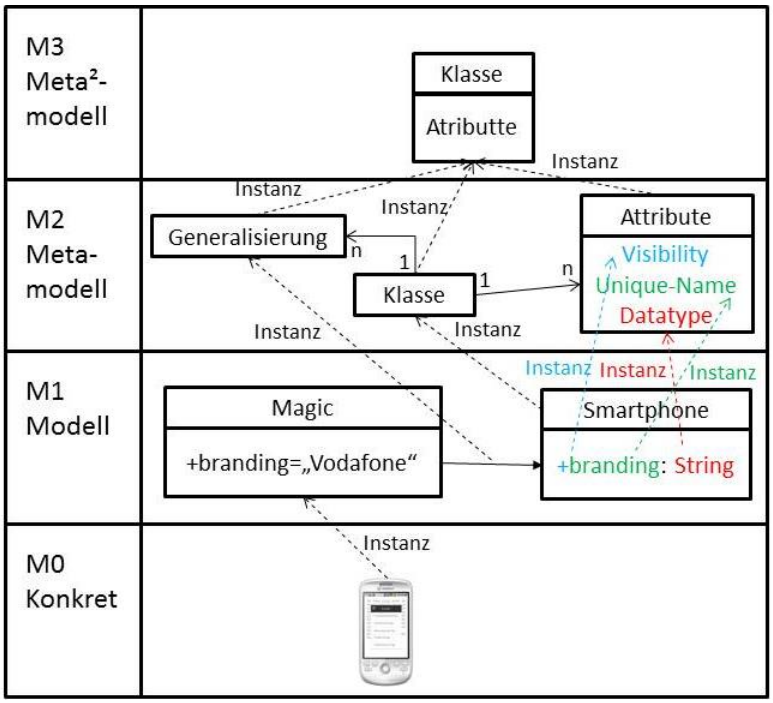


Abbildung 1 zeigt ein Modell mit 4 Ebenen, das den OMG Standard MOF widerspiegelt.

Wie die Abbildung bereits zeigt, ist das konkrete Modell in Ebene M0 (Konkret) angesiedelt. Diese Ebene beinhaltet unterschiedliche reelle Objekte, die von Endbenutzer verwendet werden können. Die Ebene M0 bietet dem Anwender konkrete Inhalte an. In diesem Fall ist es ein Smartphone mit dem Namen „Magic“ vom Mobilfunkanbieter „Vodafon“. Die Ebene M1 (Modell) beinhaltet Modelle, die Daten der M0 Ebene definieren können. Im Falle der Abbildung 1 handelt es sich um eine Instanz „Magic“ und dem generellen Objekt „Smartphone“. Die Ebene M2 (Metamodell) beinhaltet Regeln, wie Modelle in der Ebene M1 aufgebaut sein müssen. Hier werden Regeln festgelegt, wie die Struktur, Elemente, Attribute, Beziehungen und Eigenschaften eines Modells in der Ebene M1 auszusehen haben. In der konkreten Abbildung 1 wird festgehalten, dass eine „Klasse“ mehrere Attribute haben kann und dass es mehrere Generalisierungs-Beziehungen zwischen Klassen geben kann. In der Ebene M3 (Meta-Metamodell oder auch Meta^{2}modell) befindet man sich auf der MOF-Ebene. Hier werden Möglichkeiten definiert, wie man Modelle in der M2 darstellen kann. In dem konkreten Beispiel wird bestimmt, dass eine „Klasse“ „Attribute“ haben kann. Es ist eine abstrakte Ebene, mit der man gleichzeitig die Ebene M2 und M3 beschreiben kann, da es sonst nie ein Ende der Metamodelle für Metamodelle geben würde. (OMG Metamodell) (UML@Work)

In der Informatik gibt es nicht nur Metamodelle, sondern auch Metadaten oder den Metainterpreter. Metadaten werden dafür verwendet um Daten mithilfe von Daten zu beschreiben. Das können zum Beispiel Metadaten zu HTML-Seiten, Bildern, Textdokumenten oder vieles mehr sein. Ebenso wird der Metainterpreter zur Beschreibung des Programminterpreter verwendet. Dies soll zeigen, dass man in der Informatik beim Begriff „meta“ immer von einer sehr ähnlichen Bedeutung ausgeht. (Informatik Duden) (Komplexität Domain-Models) (Metamodeling Foundation)

Die Idee des „Meta”-Begriffs kommt nicht nur in der Informatik vor, sondern wird auch in anderen Wissenschaften verwendet. Seit dem 17. Jahrhundert ist eines der grundlegenden Themen der Philosophie sich mit Wirklichkeit und Bewusstsein auseinanderzusetzten. Eine „Meta“-Idee in der Philosophie taucht zum Beispiel bei dem Philosophen George Berkley auf, der den weltberühmten Satz von René Descartes „ego cogito, ergo sum“ (deutsch: „Ich denke, also bin ich“ Zitat von René Descartes aus seinem Werk „Meditationes de prima philosophia“ aus dem Jahr 1641) hinterfragte. Er stellte sich die Frage, wie Erkenntnis von der Welt möglich ist, denn woher weiß man überhaupt, dass die Welt auch Wirklichkeit ist? Zu diesem Thema verfasste Jostein Gaarden ein sehr berühmtes Buch mit dem Titel: „Sofies Welt“. In diesem Buch gibt ein Philosophielehrer einem jungen Mädchen Unterricht in Philosophiegeschichte. Im weiteren Verlauf des Buches stellt sich heraus, dass Sofies Welt Teil einer Geschichte ist, die ein UN Soldat als Buch für seine Tochter Hilde schrieb.

„Aber es ist denkbar, daß ein ganz anderer Autor irgendwo sitzt und ein Buch schreibt, das von diesem UN-Major Albert Knag handelt, der für seine Tochter Hilde ein Buch schreibt. Dieses Buch handelt von einem gewissen Alberto Knox, der plötzlich anfängt, Sofie Amundsen im Kløverveien 3 bescheidene Philosophielektionen zu schicken.“ (Zitat von Jostein Gaarder (Sofies Welt) Seite 422 5. Absatz). Dieses Zitat kann man auch in die Abbildung 1 hineinlegen. In diesem Fall ist Sofies Welt die Instanz (M0), die Geschichte - geschrieben von einem Soldaten in einem Buch - das Modell der Geschichte (M1) und das Buch - geschrieben von Jostein Gaarder - wäre dann das Metamodell (M2). Einfach ausgedrückt kann man auch sagen, die Welt wurde von einer Geschichte modelliert und die Geschichte selbst von einer Metageschichte modelliert. (Sofies Welt) (Conceptual Modeling)

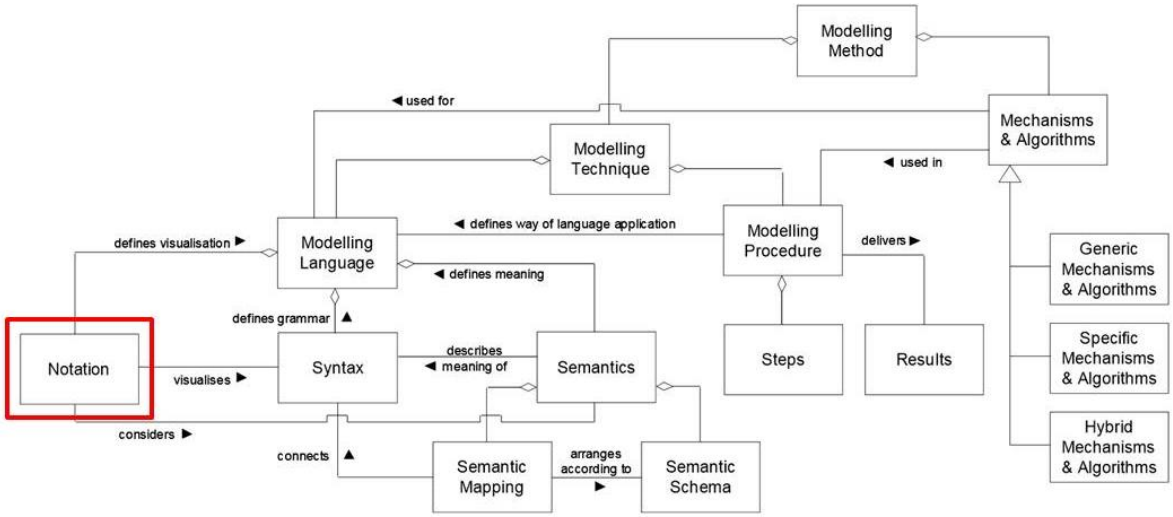


Abbildung 2 zeigt ein Modell zum Metamodellkonzept. (Zitat Kapitel 2 (Metamodelling Platforms) Seite 3 Fig. 2)

Die Abbildung 2 bildet die Komponenten von Modellierungsmethoden ab. Die Abbildung 2 zeigt alle wichtigen Komponenten, die bei einer Modellierungsmethode bedacht und abgedeckt werden müssen. Eine Modellierungsmethode (Modelling Method) besteht aus einer Modellierungstechnik (Modelling Technique) und Mechanismen und Algorithmen (Mechanisms & Algorithms). Die Modellierungstechnik besteht aus Modellierungssprache (Modelling Language) und Modellierungsprozedur (Modelling Procedure). Die Modellierungssprache besteht aus der Notation (Notation), Syntax (Syntax) und Semantik (Semantics). Diese Arbeit beschäftig sich mit dem Bereich der Notation, der in der Abbildung 2 mit einem roten Viereck markiert ist.

2.3. Einsatzgebiete von Modellen

In der Informatik kommen viele verschiedene Arten von Modellen zum Einsatz. Diese können zum Beispiel zur Softwareentwicklung oder auch um Geschäftsprozesse in einem Unternehmen abzubilden eingesetzt werden. Ein Geschäftsprozess stellt eine Reihe von Tätigkeiten dar, die in einer bestimmten Reihenfolge ausgeführt werden, um ein gewisses Ziel zu erreichen. Wenn man Geschäftsprozesse grafisch darstellt, wird von der Geschäftsprozessmodellierung[3] gesprochen. In der Geschäftsprozessmodellierung werden Geschäftsprozesse und deren Ablauf modelliert. Dies geschieht um beispielsweise Geschäftsprozesse in einem Unternehmen zu dokumentieren. Dies kann

[3] Ein Beispiel ist die Modellierungssprache BPMN die in Kapitel 3.2.2 näher erklärt wird.

dem Unternehmen unter anderem dabei helfen neue Mitarbeiter leichter einzuschulen, Informationsverlust vorzubeugen oder einfach bessere Kenntnis über unternehmensinterne Abläufe zu bekommen. In vielen Unternehmen wird die Geschäftsprozessmodellierung auch für potentielle Geschäftsprozessoptimierung verwendet. Die Geschäftsprozessmodellierung soll zum Beispiel doppelte Arbeitsschritte, Möglichkeiten schnellere und effizientere Arbeitsweisen oder straffere Arbeitsabläufe identifizieren, damit ein Arbeitsvorgang optimiert werden kann. Weiters kann die Geschäftsprozessmodellierung auch bei der Automatisierung von Prozessen oder Prozeduren helfen. (OMG BPMN)

Ein Beispiel für einen Geschäftsprozess stellt folgende Abbildung dar:

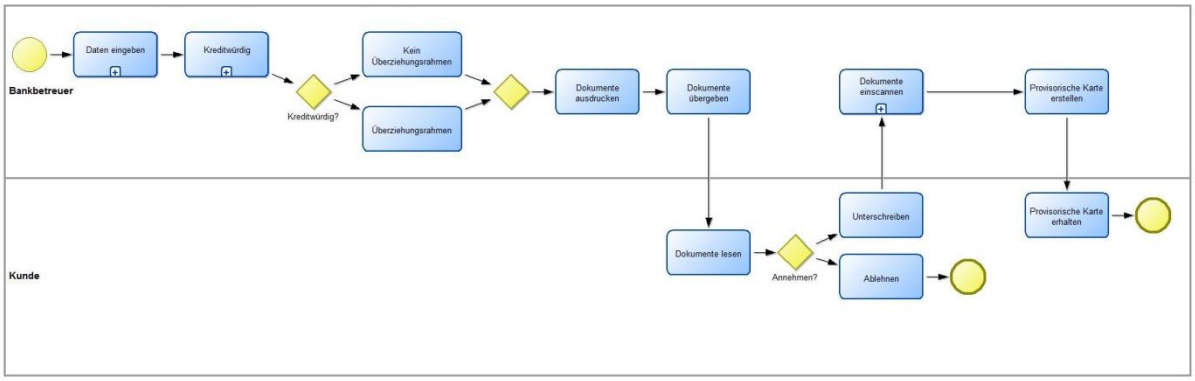


Abbildung 3 zeigt einen Geschäftsprozess zwischen einem Kunden und einem Bankbetreuer, der die Anlage eines Bankkontos in einer Filiale darstellen soll.

Dieses Beispiel zeigt den Arbeitsablauf, den ein Bankbetreuer durchlaufen muss, um einem Kunden ein Bankkonto einzurichten. Mehr Informationen zu der Notation folgen in Kapitel 3.2.2.

Modelle können auch zur Unterstützung von Softwareentwicklung eingesetzt werden. Bei der Softwareentwicklung können Modelle der Modellierungssprache Unified Modeling Language (UML) zum Einsatz kommen. Die UML 2.0 umfasst mehrere unterschiedliche Modellarten:

- Strukturmodellierung
    - Klassendiagramm
    - Objektdiagramm
    - Paketdiagramm
    - Komponentendiagramm
    - Kompositionsstrukturdiagramm
    - Verteilungsdiagramm
- Verhaltensmodellierung
    - Anwendungsfalldiagramm
    - Aktivitätsdiagramm
    - Zustandsdiagramm
    - Sequenzdiagramm
    - Kommunikationsdiagramm
    - Zeitverlaufsdiagramm
    - Interaktionsübersichtsdiagramm

Mit einem Klassendiagramm kann man in der objektorientierten Programmierung Klassen und deren Beziehungen darstellen. Bei komplexen Klassenstrukturen kann man so einfacher Beziehungen und Vererbung zwischen den Klassen erkennen. Das Anwendungsfalldiagramm zeigt Anwendungsfälle auf, die von einem Programm oder System abgedeckt werden müssen. Es spiegelt so zu sagen Aufgaben wider, die erledigt werden müssen, und stellt Ergebnisse dar, die erzielt werden sollen. Beim Informationsflussdiagramm wird dargestellt, welches System, welches Programm oder welches Teilprogramm zu welchem Zeitpunkt Zugriff auf welche Informationen hat. Mit Hilfe dieses Modells weiß ein Programmierer immer, auf welche Informationen er bei seinem Teilprogramm Zugriff hat. Dies kann vor allem bei großen Projekten enorm wichtig sein. (UML@Work)

Das Einsatzgebiet von Modellen ist breit gefächert und durch die großen Einsatzmöglichkeiten sollten Modelle auf kleinen Devices gut erkennbar dargestellt werden.

2.4. Open Model Initiative

Die „Open Model Initiative“ wurde im Jahr 2008 ins Leben gerufen und wird auf der Homepage mit dem treffenden Slogan „Models for everyone” (Zitat Kapitel 1.1 (Open Model Feasibility Study)), was übersetzt „Modelle für jedermann“ heißt, beschrieben. Die Initiative soll eine Community ins Leben rufen, die sich mit dem „Erstellen, Instandhalten, Bearbeiten, Verteilen und Analysieren von Modellen“(Zitat Kapitel 1.1 (Open Models Feasibility Study)) beschäftigt. Die Initiative soll den Mitgliedern helfen, Modelle schneller zu designen und bereits vorhandene Modelle wiederverwenden zu können. Die Community unterstützt durch diverse online Portale den Benutzer im Umgang mit Modellierungssprachen und Modellierungstools. Die Community stellt ein eigenes Open Models Wiki und ein Online Forum inklusive Support zur Verfügung. Die Open Models Initiative unterstützt auch Projekte, die sich mit neuen Modellierungsstandards befassen, und hilft neue Modellierungssprachen zu entwickeln. Von einigen bereits verwirklichten Projekte, wie zum Beispiel i*[4], MeLCa[5] oder SDbD[6], werden Modellierungstools von unterschiedlichen Modellierungssprachen auf der Homepage angeboten. (Open Models Resources)

2.4.1. Open Source und Open Models

Die Initiative wurde mit dem Wort „Open“ (auf Deutsch „offen“) versehen, da man offen und zugänglich für jedermann sein möchte. Das Wort „open“ soll in diesem Zusammenhang auch an „Open Source“ erinnern. Die „Open Source Initiative“ wurde im Jahr 1998 ins Leben gerufen und widmet sich der Förderung von Open-Source-Software. (Open Source Initiative) Auf der offiziellen Homepage der Open Source Initiative kann man 10 Kriterien einlesen, die erfüllt sein müssen, um als „Open Source“ zu gelten. Es genügt nicht, den Source Code einfach nur zugänglich zu machen:

1. Freie „Redistribution“
   betrifft die Verteilung der Software: man kann die Software frei weitergeben und auch als Teil einer eigenen Software weiter verkaufen. Dies muss aber in den Lizenzbedingungen festgehalten werden und es dürfen auch keine Entgelte dafür verlangt werden.

[4] I star
[5] Methods for designing large scale collaborative processes
[6] Semantic Database Design

2. Source Code
   Der Source Code muss frei und in kompilierter Form verfügbar sein.
3. Abgeleitete Arbeit
   Die Lizenz muss dem Benutzer eine Modifikation erlauben und das Programm muss unter denselben Lizenzbedingungen weiterverwendbar sein.
4. Integrität des Autors des Source Codes
   Man muss, falls in der Lizenz festgehalten, modifizierte Versionen des Programms mit einer eigenen Versionsnummer versehen. Weiters sollte der Benutzer die Möglichkeit haben die unmodifizierte Version zu erhalten.
5. Keine Diskriminierung gegen Personen oder Gruppen
   Die Lizenz darf keine Menschen diskriminieren.
6. Keine Diskriminierung gegen Länder oder Unternehmen
   Man darf niemanden aufgrund seines Standorts oder seiner Beschäftigung für ein Unternehmen ausgrenzen.
7. Distribution der Lizenz
   Alle Rechte müssen auch nach einer wiederholten Distribution noch vorhanden sein und es soll nicht nach einer zusätzlichen Lizenz verlangt werden.
8. Die Lizenz darf nicht an ein Produkt gebunden sein
   Alle Rechte der Software dürfen nicht von anderen Produkten/Softwarepaketen abhängen.
9. Die Lizenz darf keine andere Software voraussetzen
   Die Software darf keine andere Software voraussetzen oder ausgrenzen.
10. Die Lizenz muss Technologieneutral sein
    Die Lizenz darf nicht an Technologien oder Schnittstellen gebunden sein.

(Open Source Definition) Diese Punkte verdeutlichen, was es mit dem Open Source Gedanken auf sich hat. Ähnlich ist auch die Grundlage für die Open Models Initiative. Natürlich können diese 10 Punkte nicht eins zu eins übernommen werden bzw. wurden diese nicht so übernommen, aber man kann eine Vorstellung davon bekommen, in welche Richtung eine solche Initiative gehen soll. In der offiziellen Machbarkeitsstudie der Open Models Initiative haben sich die Autoren mit den Vor- und Nachteilen von Open Source auseinander gesetzt und konnten mithilfe von (Hang und Hohensohn 2003) und (Ven et al. 2008) herausarbeiten, dass einige der Vorteile von der freien Verfügbarkeit des Source Codes stammen. Ein weiterer wesentlicher Vorteil ist die

Tatsache, dass mehrere Personen gleichzeitig an dem Source Code arbeiten können, diese Arbeit somit kooperativ und distributiv durchgeführt wird. Im Kapitel 1.2.3 (Open Models Feasibility Study) wird auf die Übertragung dieser Punkte vom Open Source Gedanken in den Open Models Gedanken bezuggenommen und man verweist hier gleich auf die Idee, dass durch kooperative Arbeit eine Art „Peer-Review" Prozess entsteht und dieser für die Erstellung von Modellen von Vorteil ist. Weiters wird auf die Wiederverwendbarkeit und Weiterentwickelbarkeit von „offenen Modellen" eingegangen, da diese Aspekte sowohl bei Open Source als auch bei Open Models zum Tragen kommen. Man ist sich dessen bewusst, dass es bei Open Source auch Nachteile gibt, jedoch sehen die Autoren der (Open Models Feasibility Study) diese nur im Softwarebereich und daher kann man diese für eine Modellinitiative hinten anstellen.

2.5. Die Wirtschaftliche Situation von Applikationen

Da mittlerweile etwa jedes 5. Mobiltelefon, das verkauft wird, ein Smartphone ist, steigt das Interesse an Applikationen für mobile Endgeräte. Viele User verwenden, zur Informationsgewinnung aus dem Internet, lieber eine auf ein Smartphone optimierte App als eine Browserseite, die für einen Desktopbildschirm ausgelegt wurde. Viele Zeitungen, soziale Netzwerke oder Händler setzen bereits auf Applikationen, um dem Smartphone-User Informationen näher zu bringen. Sei es über die neuesten Lebensmittelangebote von einem Händler, wie etwa die „Billa-App" oder die neuesten Nachrichten über die „derStandard App". Auch soziale Netzwerke, wie etwa Facebook oder Xing, entwickelten eine eigene App, um dem User die Inhalte optimiert für kleine Device anbieten zu können.

Die Applikationen werden von unterschiedlichen Softwareherstellern entwickelt, programmiert und danach in Applikation Stores (App-Stores) vertrieben. Jedes Smartphone-Betriebssystem hat seine eigenen Applikationen, die in unterschiedlichen Programmiersprachen programmiert werden. Da die Applikationen für jedes dieser Betriebssysteme eigens entwickelt werden, sind sie auch in unterschiedlichen Stores für den Benutzer zugänglich.

Die meisten der Applikationen kosten zwischen 0 und 5 € bzw. $. Gemäß dem Marktforschungsunternehmen Gartner, Inc. wurden im Jahr 2013 etwa 102 Milliarden (Mrd.) Applikationen von Smartphone-Benutzern heruntergeladen. Da viele dieser Apps

frei verfügbar sind (92,876 Mrd.), wurden „nur“ für 9,186 Mrd. Apps in den Applikation-Stores bezahlt. (Gartner Forecast Apps) Diese 9,186 Mrd. Apps. haben für einen Umsatz in der Höhe von 26 Mrd. $ (etwa 20,43 Mrd €) gesorgt. (Guardian Revenue Apps) (Gartner Revenue Apps) Die folgende Abbildung zeigt, wie die Verkaufszahlen im Jahr 2012 und 2013 ausgesehen haben und wie die Prognosen für die Jahre 2014 bis 2017 aussehen:

	2012	2013	2014	2015	2016	2017
Free Downloads	57.331	92.876	127.704	167.054	211.313	253.914
Paid-for Downloads	6.654	9.186	11.105	12.574	13.488	14.778
Total Downloads	63.985	102.062	138.809	179.628	224.801	268.692
Free Downloads %	89,6%	91,0%	92,0%	93,0%	94,0%	94,5%

Tabelle 1 zeigt einen Überblick über alle freien und kostenpflichtigen Applikationsdownloads in den Jahren 2012 und 2013. Die Daten von 2014 bis 2017 sind Prognosen, die im Jahr 2013 erstellt wurden. Die Zahlen sind in Millionen Stück angegeben. (Zitat von Gartner Inc. (Gartner Forecast Apps) „Table 1. Mobile App Store Downloads, Worldwide“)

Wie man in der letzten Zeile der Tabelle 1 erkennen kann, geht Gartner davon aus, dass sich der Anteil an Gratis-Downloads in den Jahren stetig erhöhen wird. Da sich aber auch die Downloadzahlen der Applikationen erhöht, steigen die Applikationsverkäufe in absoluten Zahlen an. Daran lässt sich ablesen, dass sich dieses Marktsegment stetig weiterentwickelt, und welch riesiges finanzielles Potential im Applikationsmarkt für Smartphones und Tablets vorhanden ist. (Guardian Revenue Apps) (Gartner Revenue Apps)

Durch diese Tatsachen kann man eindeutig erkennen, dass in Zukunft mobiles Arbeiten mit Smartphones an Attraktivität gewinnen wird. Die Applikationsverkäufe werden in Zukunft ansteigen. Bereits jetzt gibt es viele „Arbeits-Applikationen“ (das sind Applikationen, die zum mobilen Arbeiten verwendet werden) und in Zukunft wird ihre Anzahl sicher stetig steigen. Aus diesem Grund hat sich der Autor dieser Arbeit mit dem Vorteilen, Nachteilen und den Problemen der Modellierung auf mobilen Geräten befasst.

Bereits 2006 haben Dejin Zhao, John Grundy und John Hosking ein Java-Programm für Nokia Mobiltelefone veröffentlicht, mit dem man modellieren kann. Damals gab es noch kaum Smartphones und es wurden Programme noch nicht über App-Stores vertrieben,

sondern direkt auf Homepages der Entwickler zum Download angeboten. Die Installation eines solchen Java-Programms wurde durch den Benutzer am Gerät selbst durchgeführt.

Heutzutage kann man, wie bereits im Kapitel 1 kurz erwähnt, theoretisch bei Android, Windows Mobile, iOS und auch Symbian eigene Applikationen auf dem Gerät installieren, die nicht in einem offiziellen Store verfügbar sind. Die Hersteller raten von einem solchen Vorgehen jedoch stark ab, da diese Applikationen keinem Review-Prozess unterzogen worden sind und man sich daher nicht sicher sein kann, dass die Applikation keine Malware, Viren, Trojaner oder sonstige schadhafte Inhalte umfasst. Beim iOS von Apple gibt es unter anderem die Möglichkeit sich die „Ad-Hoc-Distribution", die für Unternehmen gedacht ist, zu Nutze zu machen, um eigene Applikationen zu installieren. Dadurch kann man Applikationen am Gerät installieren, die nicht für jedermann zugänglich sein sollen, oder die nicht am Review-Prozess von Apple teilgenommen haben, oder den Review-Prozess nicht überstanden haben. (Computerwoche iPhone) Bei den anderen drei Herstellern ist die Installation von App-Store fremden Applikationen einfacher und nicht so aufwendig. Bei Android kann man alle funktionsfähigen Applikationen auf dem Endgerät installieren, egal woher man diese bekommt (App-Store, Homepages, SD-Karten …). Natürlich gibt es auch hier ein geschlossenes System, das von Google angeboten wird: Market[7]. Bei dem Betriebssystemvorgänger von Windows Phone 7, also zum Beispiel Windows Mobile 6, 6.1 und 6.5, konnten einfache C++ oder .Net Anwendungen auf den Geräten installiert werden. Hierfür war keinerlei Review-Prozess oder ein Store notwendig. Entwickler hatten ihre Applikation frei oder kommerziell auf Homepages im Internet angeboten. Bei Windows Phone 7 gibt es zwar auch einen Marketplace[8], aber man kann auch hier für sein Unternehmen eigene Applikationen programmieren und auf den Geräten installieren. (TecChannel Windows Phone 7)

2.6. Wirtschaftliche Situation von Smartphones

Die aktuelle Situation am Markt der Smartphones ist recht angespannt. Es gibt weltweit sehr viele Hersteller von Endgeräten, wie zum Beispiel Apple, HTC, Huawei, LG,

[7] Der Market ist bei Android der Ort, wo die bereits geprüften Applikationen angeboten werden.
[8] Der Marketplace ist bei Microsoft der Ort, wo die Applikationen angeboten werden. Wie beim App-Store werden hier alle Applikationen überprüft und danach für alle Benutzer freigeschalten.

Motorola, Nokia, RIM, Samsung, Sony Ericsson und ZTE, um die größten 10 zu nennen, die alle eine Vormachtstellung erreichen wollen. All diese Hersteller versuchen möglichst viele Geräte abzusetzen und momentan herrscht ein erbitterter Kampf um Patente und Lizenzabkommen. Wie Microsoft auf seinem offiziellen Blog mitgeteilt hat, haben sie nun 10 Lizenzabkommen mit Herstellern von Android Smartphones und Tablets. (Microsoft Patente)

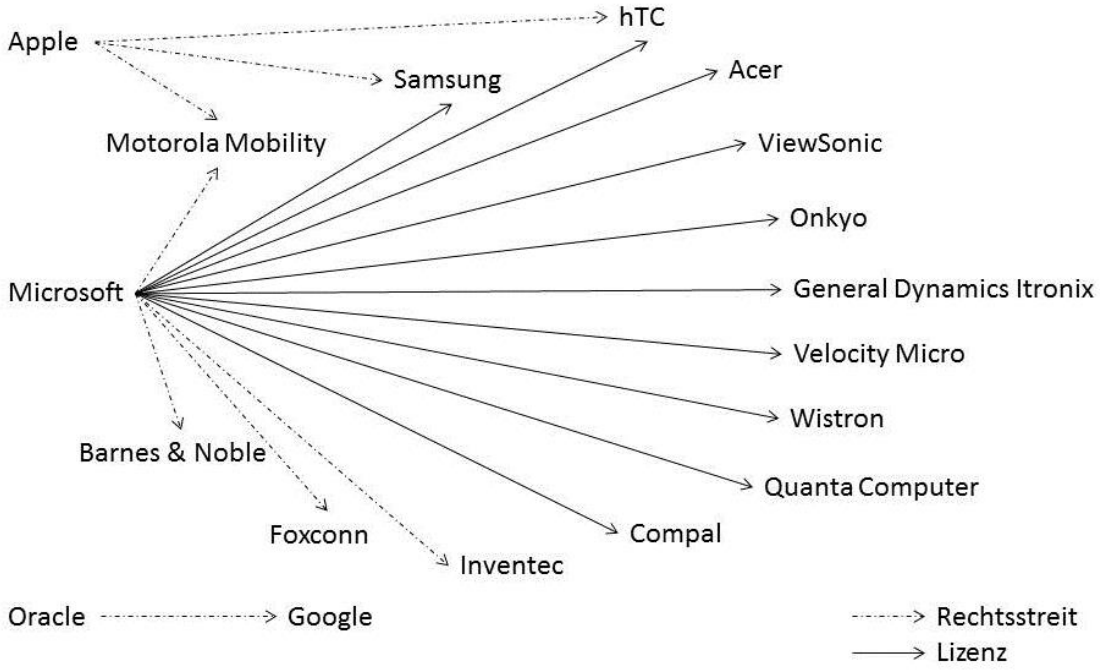


Abbildung 4 zeigt, welche Firmen Lizenzgebühren an Microsoft für die Nutzung von Patenten bezahlen. (Zitat (MS Patente Grafik))

Wie in der Abbildung 4 ersichtlich ist, stellt Microsoft sehr vielen Herstellern seine Patente zur Verfügung. Damit bekommt das Unternehmen einen Anteil des Verkaufserlöses von jedem Gerät, das von diesen Firmen mit dem Android Betriebssystem verkauft wird. (Microsoft Patente)

Im Jahr 2012 hatte auch Google erkannt, dass es am Markt für Smartphones und Tablets sehr hilfreich sein kann, Patente zu besitzen, und hat die Firma Motorola für 12,5 Milliarden US-Dollar gekauft. Dies umfasste das komplette Unternehmen inklusive aller (Mobilfunk)Patente. Im Jahr 2014, also ungefähr 2 Jahre später, hat Google Motorola an Lenovo um 2,91 Milliarden US-Dollar weiterverkauft. Jedoch hat man nur das

Unternehmen nicht aber die Patente verkauft. Anders gerechnet kann man sagen, dass sich Google die Patente der Firma Motorola 9,59 Milliarden US-Dollar kosten hat lassen. Google Chef Larry Page erklärt, dass die Patente bei Google bleiben „um gesamte Android-Ökosystem zu verteidigen. " (Zitat von Larry Page 2 Absatz (Lenovo kauft Motorola) Hier kann man erkennen, wie umstritten und gleichzeitig auch wichtig die Patentrechte für die Unternehmen in diesem Geschäft sind.

All diese Unternehmen sehen ein sehr großes Potential in diesem Markt und kämpfen daher verbissen um eine Vormachtstellung. Steve Jobs, der schon verstorbene Mitbegründer der Firma Apple, hat dies einmal sehr deutlich zum Ausdruck gebracht: „Wenn es sein muss, werde ich das bis an mein Lebensende und mit jedem Penny der 40 Milliarden Dollar von Apple, die auf der Bank liegen, richtigstellen. Ich werde Android zerstören, denn es ist ein geklautes Produkt. Ich bin bereit, dafür einen thermonuklearen Krieg anzufangen." (Zitat: Steve Jobs: Die autorisierte Biografie des Apple-Gründers S. 351 Absatz 5) (Steve Jobs Biografie)

Die folgenden Abbildungen veranschaulichen die Absatzzahlen von Smartphones nach deren Betriebssystemen:

Betriebssystem	2013 Stück	2013 Marktanteil (%)	2012 Stück	2012 Marktanteil (%)
Android	758.719,90	78,4	451.621,00	66,4
iOS	150.785,90	15,6	130.133,20	19,1
Microsoft	30.842,90	3,2	16.940,70	2,5
BlackBerry	18.605,90	1,9	34.210,30	5,0
Andere OS	8.821,20	0,9	47.203,00	6,9
Total	**967.775,80**	**100,0**	**680.108,20**	**100,0**

Tabelle 2 zeigt die Verkaufszahlen von Smartphones nach Betriebssystemen in den Jahren 2012 und 2013. Gartner veröffentlichte diese Zahlen im Februar 2014. (Zitat von Gartner Inc. (Gartner sales 2014) „Table 3 Worldwide Smartphone Sales to End Users by Operating System in 2013")

In der Tabelle 2 werden die Verkaufszahlen aus den Jahren 2012 und 2013 gezeigt. Man kann erkennen, dass Geräte mit Android und iOS mit Abstand am meisten verkauft werden. Natürlich sind die Verkaufszahlen des Betriebssystems Android viel höher als die von iOS, jedoch wurden zu diesem Zeitpunkt nur vier verschiedene Geräte mit dem iOS Betriebssystem verkauft. Im Gegensatz dazu gibt es viel mehr Endgeräte mit dem Android System.

Da man auch davon ausgehen kann, dass, wie schon im Kapitel 2.5. beschrieben, die Verkaufszahlen der Applikationen immer weiter steigen werden, werden Unternehmen um die Position des Marktführers, im Smartphonebereich, heftig konkurrieren. Wie die Tabelle 2 erkennen lässt, steigen die Absatzzahlen von Jahr zu Jahr und noch ist laut Prognosen von Gartner keine Ende in Sicht. (Smartphones Sales 2014) (Gartner Forecast Apps) (derStandard US-Markt Smartphones)

Die Frage, die sich nun für alle stellen sollte, ist: „Auf welchem Smartphone Betriebssystem sollte man seine Applikation programmieren?“ Die meisten Applikationen erscheinen auf Android und iOS und sind somit auch nur für diese Betriebssysteme verfügbar. Durch die Zusammenarbeit von Nokia und Mircosoft kann man durchaus auch mit dem Windows Phone Betriebssystem in Zukunft rechnen. (Gartner Smartphones 2010-2015) Man kann auch sehen, dass sich die Verkaufszahlen von Windows Phone Geräten von 2012 auf 2013 verdoppelt haben, der allgemeine Marktanteil jedoch immer noch auf vergleichsweise niedrigen 3,2% liegt.(Gartner sales 2014) Gartner prognostiziert, dass Android seine Vormachtstellung verteidigen kann, aber durch die Zusammenarbeit von Windows und Nokia wird es mit dem Symbian[9] Betriebssystem bergab und mit Windows Phone, dem Betriebssystem von Microsoft, bergauf gehen. In der Tabelle 2 kann man sehen, dass die Marktanteile von „Anderen Betriebssystemen“ von 2012 auf 2013 stark gesunken sind. Hier ist unter anderem das Symbian Betriebssystem mitgezählt worden. Man sieht, dass die Zahlen hier stark gesunken sind.

Daher sollte man sich bei der Entwicklung einer Applikation die Frage zwischen iOS und Android stellen.

[9] Symbian ist das Betriebssystem, das aktuell auf den meisten Nokia Smartphones installiert ist.

3. Modell-Notation für Smartphones

Wie auch bereits in Kapitel 1 erwähnt, muss man sich bei der Notation und Darstellung auf sehr kleine Screens beschränken. Die Smartphone Screens wurden in den letzten Jahren zwar etwas größer, jedoch gibt es kaum welche, deren Screens größer als 3“ bis 5“sind. In Abbildung 5 werden drei Smartphones in den Größen 3“, 4“ und 5“ gezeigt, mit demselben RDF Graphen angepasst auf die jeweilige Bildschirmgröße.

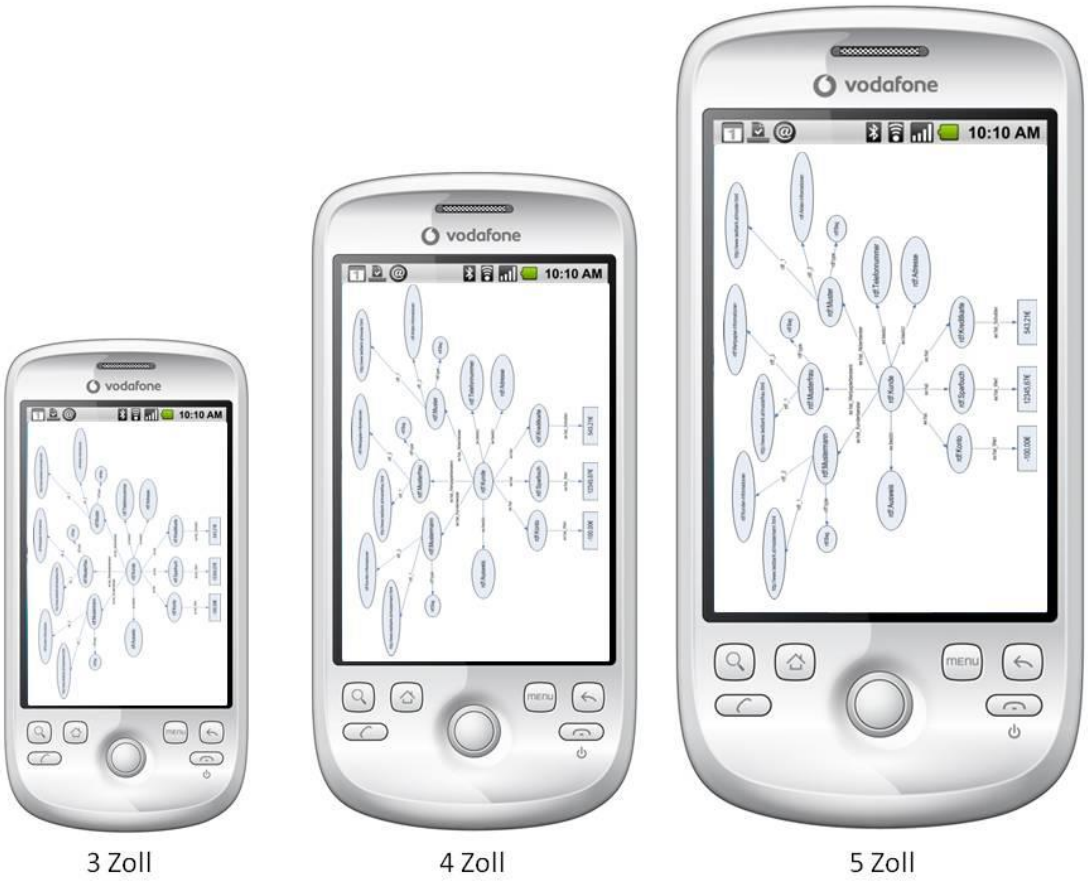


Abbildung 5 zeigt 3 Smartphones mit demselben RDF Graphen (Abbildung des Smartphones Zitat (HTC Magic))

Ein durchschnittliches mobiles Device kann jedes gängige Bildformat anzeigen. Die Frage, der in diesem Kapitel nachgegangen werden soll, ist jedoch, ob es Probleme bei der Übersichtlichkeit der Modelle als Bilder geben kann. Hier muss eine adäquate Möglichkeit gefunden werden, um die Darstellung auf kleinen Screens optimal zu

gewährleisten. Dieses Problem kann mit zwei unterschiedlichen Darstellungsmöglichkeiten auf Smartphones gelöst werden:

3.1. Wie kann man Modelle auf Smartphones darstellen?

Ein Modell kann auf einem Smartphone durch einen Text, eine Notation oder ein Bild dargestellt werden. Durch die Multitouch Technologie kann man Bilder mit einer hohen Pixelanzahl vergrößern, um Elemente näher zu betrachten. Die Pixeldichte auf einem durchschnittlichen Computermonitor ist etwa 100 pixel per inch (ppi). Bei 100 ppi entspricht jeder Bildpunkt 0,3 Millimeter (mm). Ein modernes Smartphone hat wesentlich mehr Pixel, damit man auf den kleinen Bildschirmen alles gut erkennen kann. Aktuell haben die meisten Smartphones eine Auflösung von 1080x1920 Pixel (Full-HD), was bei 4,7“ Bildschirmdiagonale 469 ppi (0,0542 mm pro Pixel) entspricht. Der Trend geht zu immer mehr Pixel und in Zukunft werden Smartphones noch hochauflösendere Bildschirme mit Auflösungen von 1440x2560 Pixel (entspricht bei 5,5“ Bildschirmdiagonale 534 ppi und 0,0476 mm pro Pixel) oder auch 3840x2160 Pixel (entspricht bei 5,5“ Bildschirmdiagonale 801 ppi und 0,0317 mm pro Pixel) bekommen. (chip Bildschirmauflösung) Die Bildschirmauflösung 3840x2160 wird auch „4K“ genannt. (Heise.de 4k) Das hat natürlich Vor- und Nachteile.

3.1.1. Textuell

Ein Modell kann nicht nur mit Hilfe von Bildern und Symbolen dargestellt, sondern auch in Worten beschrieben werden. Eine textuelle Beschreibung könnte in Stichworten oder in Form von Textbausteinen geschehen.

Es gibt in der Informatik bereits Auszeichnungssprachen, die sowohl in textueller als auch in graphischer Art dargestellt werden: Resource Description Framework (RDF). Die Sprache RDF wird zur Beschreibung von Ressourcen benutzt und kann sowohl als Graph als auch in Form von Tag-Elementen (in Form eines XML-Dokuments) dargestellt werden. Der Graph ist für den Menschen einfacher zu lesen und auch übersichtlicher im Gegensatz zu den XML-Tags. (Semantik Web) Ein Beispiel für solch einen Graphen und ein RDF-Dokument ist in Abbildung 6 und im folgenden XML-Code zu sehen:

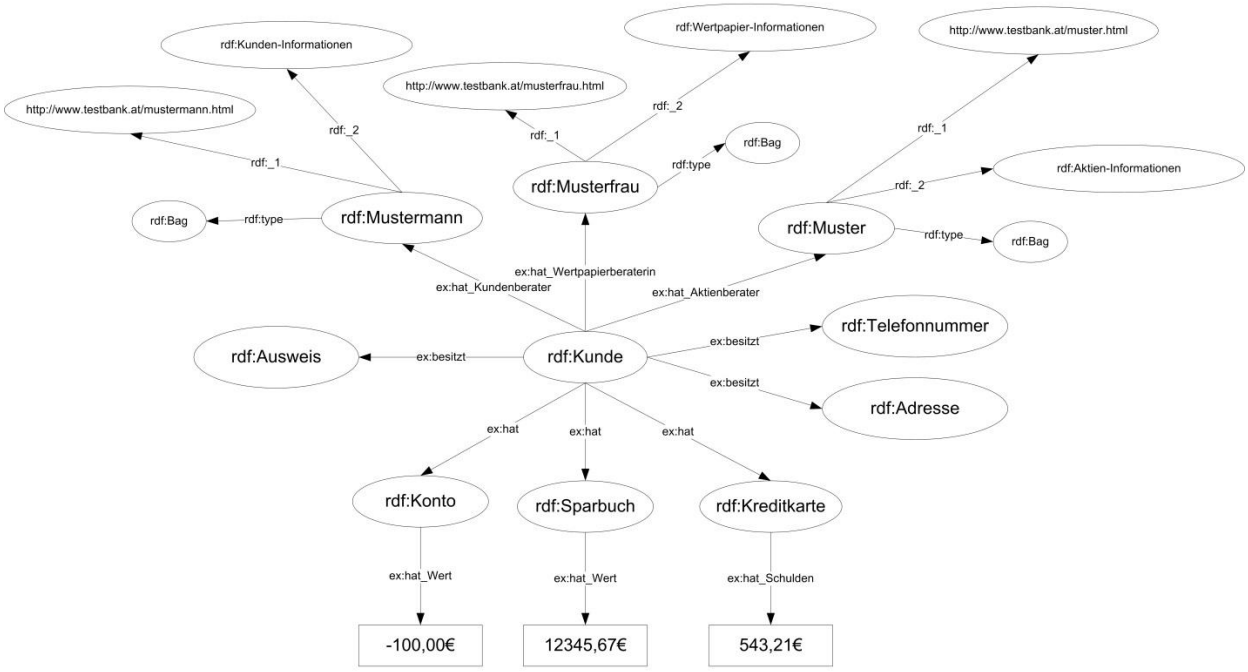


Abbildung 6 zeigt einen RDF Graphen

In dem Beispiel aus Abbildung 6 geht es um den Kunden einer Bank. Dieser Kunde hat ein Konto, eine Kreditkarte und ein Sparbuch bei einer Bank. Der Graph zeigt den jeweiligen finanziellen Status der Finanzressourcen. Man kann auch leicht erkennen, dass der Kunde einige persönliche Informationen bei der Bank hinterlegt hat und mehrere persönliche Berater in finanziellen Angelegenheiten hat. In XML-Schreibweise würde solch ein Graph wie folgt dargestellt:

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ex="http://testbank.at/example#">
  <rdf:Description rdf:about="rdf:Kunde">
    <ex:hat>
      <rdf:Description rdf:about="rdf:Konto">
        <ex:hat_Wert rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
          -100,00€
        </ex:hat_Wert>
      </rdf:Description>
    </ex:hat>
    <ex:hat>
      <rdf:Description rdf:about="rdf:Sparbuch">
        <ex:hat_Wert rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
          12345,67€
        </ex:hat_Wert>
      </rdf:Description>
    </ex:hat>
```

```
    <ex:hat>
      <rdf:Description rdf:about="rdf:Kreditkarte">
        <ex:hat_Schulden rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
          543,21€
        </ex:hat_Schulden>
      </rdf:Description>
    </ex:hat>
    <ex:besitzt rdf:resource="rdf:Ausweis"/>
    <ex:besitzt rdf:resource="rdf:Telefonnummer"/>
    <ex:besitzt rdf:resource="rdf:Adresse"/>
    <ex:hat_Kundenberater>
      <rdf:Bag rdf:about="rdf:Mustermann">
        <rdf:_1 resource="http://www.testbank.at/mustermann.html"/>
        <rdf:_2 resource="rdf:Kunden-Informationen"/>
      </rdf:Bag>
    </ex:hat_Kundenberater>
    <ex:hat_Wertpapierberaterin>
      <rdf:Bag rdf:about="rdf:Musterfrau">
        <rdf:_1 resource="http://www.testbank.at/musterfrau.html"/>
        <rdf:_2 resource="rdf:Wertpapier-Informationen"/>
      </rdf:Bag>
    </ex:hat_Wertpapierberaterin>
    <ex:hat_Aktienberater>
      <rdf:Bag rdf:about="rdf:Muster">
        <rdf:_1 resource="http://www.testbank.at/muster.html"/>
        <rdf:_2 resource="rdf:Aktien-Informationen"/>
      </rdf:Bag>
    </ex:hat_Aktienberater>
  </rdf:Description>
</rdf:RDF>
```

Wie man sehr leicht sehen kann, ist die XML-Schreibweise für ein relativ kleines Modell mit nur 16 Ressource-Elementen schon sehr komplex und aufwendig. Es ist ein relativ langer Code für so wenige Elemente und für den Menschen nur sehr schwer lesbar. Die graphische Darstellung des RDF-Graphen aus Abbildung 6 ist jedoch „(…) leicht verständlich und auch präzise, aber sie ist für die Verarbeitung der Daten in Computersysteme denkbar ungeeignet. Zudem können auch Menschen nur sehr kleine Graphen mühelos verstehen – praktisch auftretende Datensätze mit Tausenden oder Millionen von Knoten und Kanten eignen sich dagegen nicht zur zeichnerischen Präsentation.“ (Zitat „Semantik Web“ S 39 f) Bei RDF wurde bereits von den Entwicklern sichergestellt, dass auch der Mensch mit der graphischen Darstellung das Modell gut verstehen kann. Wie man jedoch in der Quelle (Semantik Web) nachlesen kann, wenn ein Modell zu groß wird, ist es in jeder Darstellung für den Menschen nicht mehr lesbar.

Also man muss sich bei jedem Modell, egal auf welchem Bildschirm, ab einer gewissen Größe bewusst sein, dass der Benutzer das Modell nicht mehr mühelos verstehen kann.

Die Beschreibung eines Modells in textueller Form mit Hilfe von XML-Tags ist auf einem kleinen Smartphone-Bildschirm nicht zu empfehlen und für den Menschen, im Gegensatz zu einer Maschine, nicht einfach lesbar. Man sollte in diesem Fall auf Stichworte oder Textbausteine zurückgreifen. Man könnte als Alternative auch an eine „Mischform" der beiden Darstellungsmöglichkeiten denken. Ein Modell könnte so vereinfacht werden und übersichtlich bleiben. Oder man verbindet Stichworte mit Symbolen um ein übersichtliches Modell zu erzeugen.

Die Form der Darstellung mit Hilfe von Textbausteinen, würde jedes Element (Symbole und auch Graphen) durch einen dieser Bausteine ersetzen. Man müsste einen Text für ein Element definieren und zusätzlich den Namen des Elements in diesen Text einfügen. Nur so kann man die Verwechslungen im Text verhindern, da die meisten Elemente in einem Modell öfter als einmal vorkommen. Um bei dem Beispiel aus Abbildung 6 zu bleiben, könnte man jedes Element mit „Element" und den Namen des Elements angeben. Der Name der Beziehung zwischen den Elementen bleibt gleich und das Element, von dem der Pfeil ausgeht wird, als erstes genannt. Die Typisierung wie etwa „type" kann mit „ist vom Type" beschrieben werden und Ressourcen, die mit „1", „2", „3", … beschrieben werden, bekommen den allgemeinen Text „hat die Ressource". Ein Ausschnitt aus dem obigen Beispiel sieht wie folgt aus:

Element Kunde hat Element Konto. Element Konto hat_Wert -100€. Element Kunde hat_Aktienberater Element Muster. Element Muster ist vom Type Bag. Element Muster hat die Ressource Aktien-Informationen. Element Muster hat die Ressource http://www.testbank.at/muster.html

3.1.2. Graphisch

Das Problem der graphischen Darstellung auf einem kleinen Bildschirm ist die Unübersichtlichkeit. Da die Darstellung von einem Modell auf zu engem Raum vom Auge nicht mehr gut erkennbar ist.

Die graphische Darstellung eines Modells ist Visualisierung oder Veranschaulichung von Daten und Texten in Bildform. Unter Visualisierung versteht man im Allgemeinen jede

Darstellung von geistigen oder verbalen Inhalten in einer visuellen Form. Hierzu zählen nicht nur Modelle, die Texte und Daten darstellen, sondern auch Fernsehwerbungen, die ein Produkt bewerben sollen, oder auch pantomimische Darstellungen. Die Visualisierung ist in der heutigen Zeit eine der beliebtesten Methoden, um Informationen zu verbreiten bzw. zu präsentieren. In jedem gängigem Textverarbeitungsprogramm oder Tabellenkalkulationsprogramm gibt es die Möglichkeit Diagramme zu erstellen, um Daten, Fakten, Informationen und Zahlen anschaulicher und schneller erkennbar zu machen. (Datenvisualisierung & DM) Während jedes Vortrages wird in der heutigen Zeit auf das Element Visualisierung gesetzt. Auch viele Entwicklungen lassen sich mittels eines Graphen viel besser darstellen, z.B. werden alle Aktienkurse in Diagrammen dargestellt, um den Verlauf einer Aktie schneller nachvollziehen zu können. Auch die Ergebnisse einer Wahl werden in Balken-, Torten-, oder Halbkreisdiagrammen dargestellt. So kann jeder interessierte Nutzer dieser Informationen auf einen Blick erkennen, wie die Ergebnisse im Verhältnis zueinander stehen. (Visualisieren & Präsentieren)

In der Informatik wird auch sehr viel mit Informationsvisualisierung gearbeitet. Dabei geht es um die Darstellung von großen Mengen an unübersichtlichen Daten und Informationen, um diese dem Benutzer besser veranschaulichen zu können. Hier wird der Fokus auf Aufbereitung der Informationen gelegt. Ganz im Gegensatz zur Informationsvisualisierung wird die Wissensvisualisierung verwendet, um Wissen weiter zugeben oder Lerninhalte aufzuarbeiten. (Semantic Web Visualizing)

Ein Modell graphisch darzustellen ist die übliche Art und Weise, wie der Mensch es besser erkennen kann. Für Maschinen ist jedoch die textuelle Darstellung besser geeignet. Die graphische Darstellung kann dem Betrachter viel Zeit ersparen und ein Modell vereinfachen. Durch das Wissen, was die einzelnen Elemente bedeuten, kann der Betrachter schnelle Schlüsse aus dem Modell ziehen. Durch die übersichtliche Darstellung der Modelle kann man Verknüpfungen, Abhängigkeiten und Vererbungen viel schneller erkennen. Die Abbildung eines einfachen Graphen in Abbildung 7 soll dies verdeutlichen:

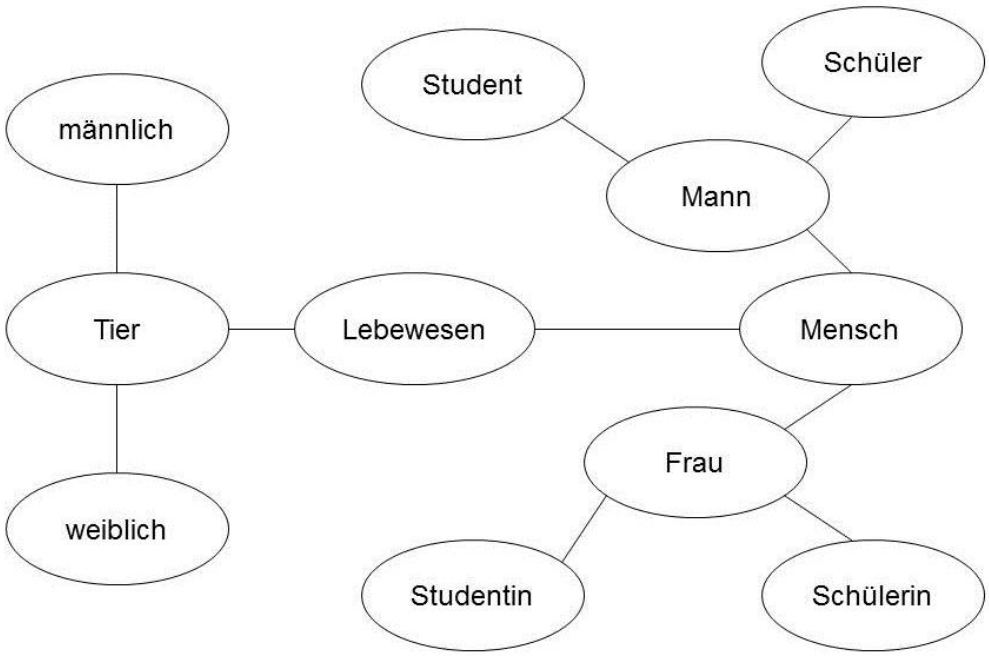


Abbildung 7 soll einen einfachen Graphen zeigen, der als Modell sehr einfach zu interpretieren ist. Es befinden sich nur eine Art von Beziehungen und nur eine Art von Element in diesem Graph.

Das Modell lässt sich sehr einfach in seiner graphischen Darstellung interpretieren. Mit Worten würde man das in etwa wie folgt beschreiben müssen:

Das Wurzelelement Lebewesen steht in Verbindung mit Mensch und Tier. Das Element Mensch steht in Verbindung mit dem Element Mann und dem Element Frau. Das Element Mann steht in Verbindung mit dem Element Student und dem Element Schüler. Das Element Frau steht in Verbindung mit dem Element Studentin und dem Element Schülerin. Das Element Tier steht in Verbindung mit dem Element männlich und dem Element weiblich.

Der in der Einleitung des Kapitels angesprochenen Unübersichtlichkeit kann auch die neue Multitouch-Technologie nicht abhelfen. Mithilfe der Multitouch-Technologie kann man ein Bild, mit Hilfe von Gesten, auf einem Smartphone vergrößern und verkleinern. Das Bild erscheint größer und die Elemente können besser erkannt werden. Unter diesem Aspekt leidet jedoch die Übersichtlichkeit eines Modells sehr stark. In Abbildung 8 wurde ein Modell mit 22 Elementen und 21 Beziehungen dargestellt. Das linke Smartphone zeigt das komplette Modell auf einem Bildschirm und das rechte Smartphone einen vergrößerten Bereich des Modells.

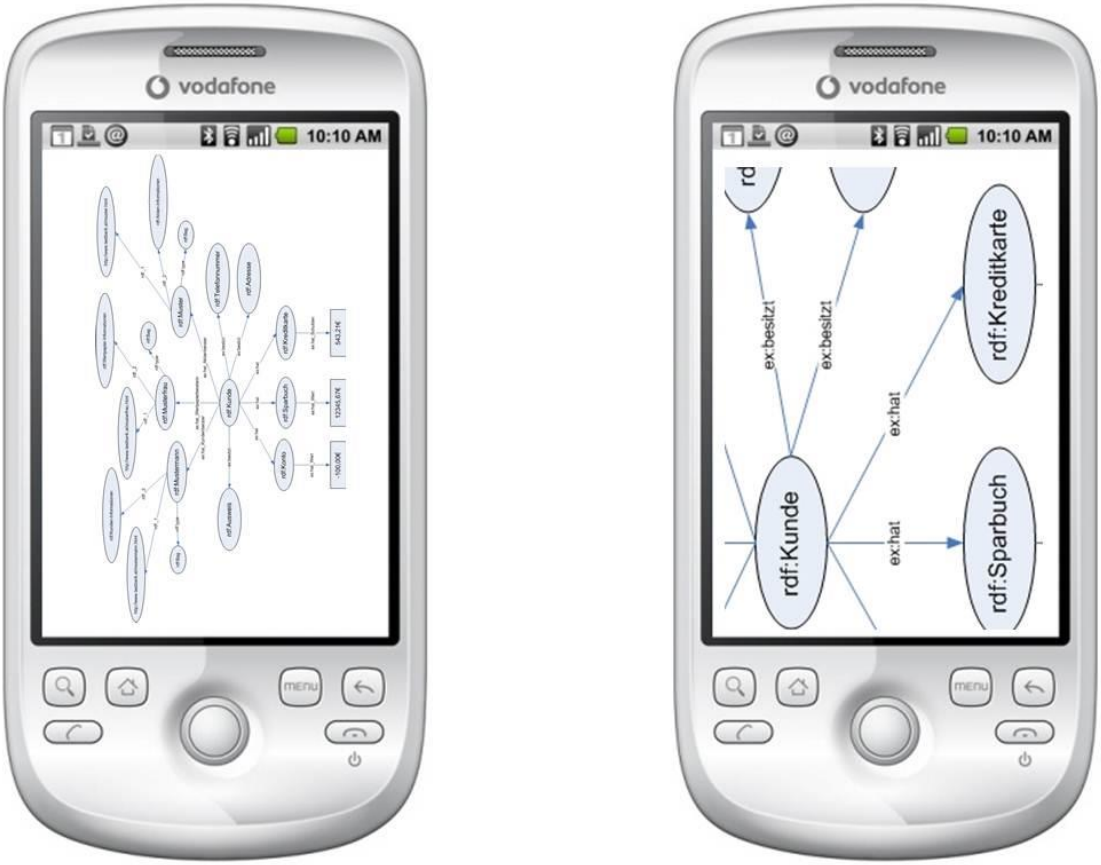


Abbildung 8 zeigt auf der linken Seite das volle RDF-Modell und auf der rechten Seite ein auf den rdf:Kunden zentriertes Modell. (Abbildung des Smartphones Zitat (HTC Magic))

Man kann am rechten Smartphone das Element gut erkennen und auch die Beschriftungen sind sehr gut lesbar. Das Problem ist jedoch, dass man nicht mehr gut erkennen kann, mit welchen anderen Elementen das vergrößerte Element interagieren soll und welche Abhängigkeiten es besitzt. Man kann von 22 Elementen nur noch 3 Elemente zentriert sehen und, um die weiteren Beziehungen zu sehen, muss das Modell hin und her geschoben werden. Dies kann bei sehr großen Modellen mühsam werden.

Bei einem Modell mit 10 Elementen kann die Multitouch-Technologie vielleicht noch aushelfen und dem Benutzer die Möglichkeit bieten, das Modell zu verstehen. Bei Modellen mit mehr als 20 Elementen wird dies immer schwieriger. Die Forscher David E. Irwin und Gregory J. Zelinsky haben in ihrer Arbeit (Eye movements and scene perception) herausgefunden, dass sich das menschliche Auge am besten auf nur fünf

Objekte einstellen kann. In ihrer Arbeit haben sie 12 Menschen eine einfache Szene mit 7 Objekten gezeigt. 6 Personen haben das Experiment mit 1, 3 oder 5 Fixierungen und die andere Hälfte der Versuchsgruppe hat es mit 3, 9 oder 15 Fixierungen durchgeführt. Die Autoren erhofften sich mit dem Experiment ein Ergebnis, das zeigt, wie viele Objekte vom menschlichen Auge erfasst werden können und wie viele Fixierungen dafür notwendig sind. Die Szene zeigte 7 Objekte in einer Kinderkrippe: Ölkreiden, eine Flasche, eine Trompete, ein Auto, ein Teddybär, eine Ente und eine Puppe. Die Testpersonen sollten je nach Gruppenzugehörigkeit 1, 3, 5, 9 oder 15 Fixierungen[10] durchführen. Diese Fixierungen wurden mitgezählt und sobald der Beobachter die gewünschte Anzahl an Fixierungen durchgeführt hat, wurde die Szene sofort verdunkelt. Die Abbildung 9 zeigt diese Prozedur:

[10] Eine Fixierung soll ein Blick auf ein Objekt / einen Punkt in der Szene sein. Die Autoren erklären das Ganze wie folgt: "Consider walking into a room that you have never been in before. The room contains more information than you can perceive in a single glace, so you direct your eyes, via rapid movements called saccades, so as to fixate objects around you." (Zitat aus (Eye movements and scene perception) Seite 882 Absatz 1) Das heißt, das Auge wandert von Objekt zu Objekt und fixiert dieses. Damit bekommt der Mensch einen Eindruck, wie es in dem Raum aussieht. Genau dasselbe soll in dem Experiment mit dieser besagten Szene passieren.

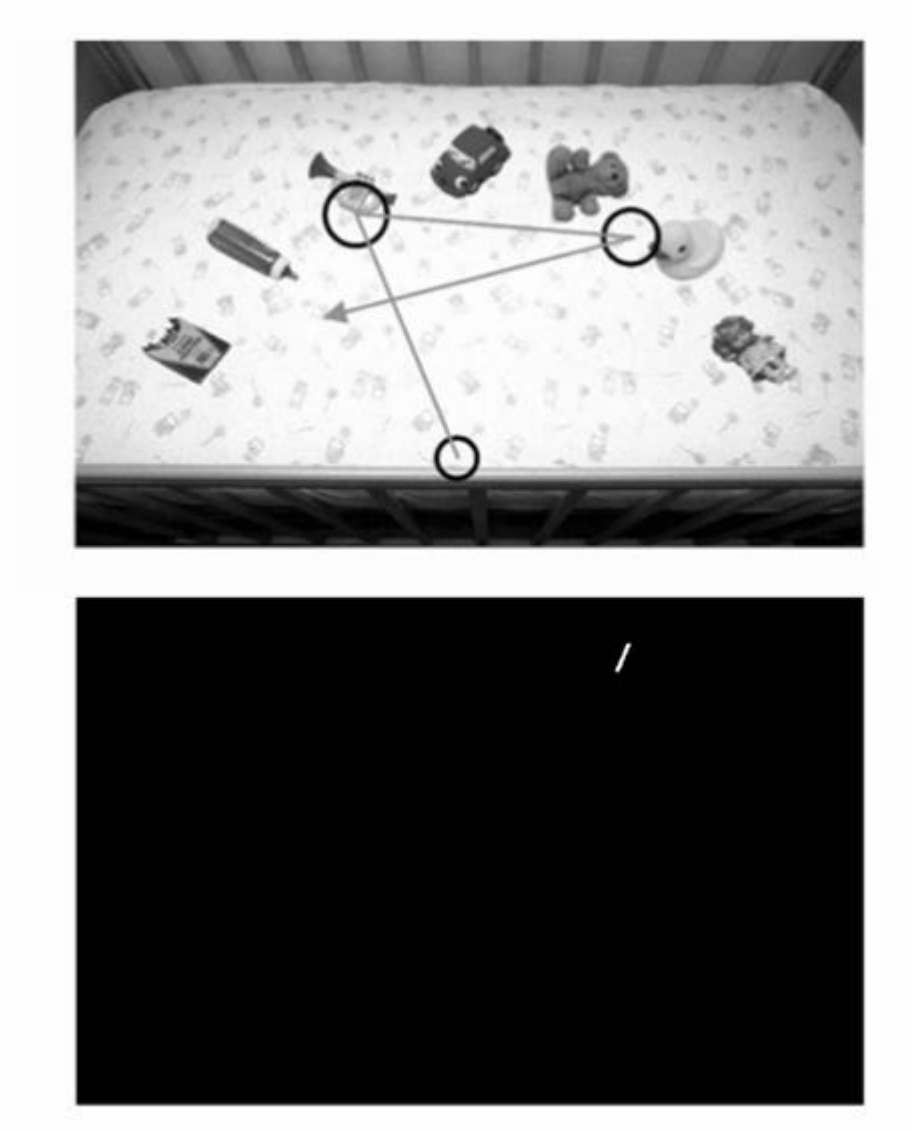


Abbildung 9 zeigt das Beispiel von 3 Fixierungen der Szene, die für das Experiment verwendet wurde. Nachdem die dritte Fixierung passiert ist, wurde die Szene für die Testperson verdunkelt. (Zitat von Abbildung 2 (Eye movements and scene perception) Seite 885)

Die Ergebnisse des Experiments wurden in einer Tabelle dargestellt:

Fixation	Condition Viewing Time (msec)	Objects Foveated
1	174	0,00
3	739	1,25
5	1360	2,68
9	2810	3,47
15	4927	5,10

Tabelle 3 veranschaulicht das Ergebnis des Experiments und man kann erkennen, dass man mit doppelter Betrachtungszeit sich nicht an doppelt so viele Objekte erinnern kann. (Zitat von Tabelle 1 (Eye movements and scene perception) Seite 885)

Die Ergebnisse zeigen, dass auch nach 15 Fixierungen die Testpersonen nur 78% der Objekte (bzw. 5 Objekte in diesem Experiment) richtig wiedergeben konnten. Das heißt, der Mensch kann sich auch nach einer relativ langen Zeit von fast 5 Sekunden (4927 Millisekunden) nur 5 der 7 Objekte merken und deren Position wiedergeben.

Fazit dieses Experiments ist, dass man den Benutzer mit Modellen auf kleinen Displays nicht überfordern sollte und daher die bereits vorhanden Modellierungssprachen adaptieren muss, da die Modelle vom Menschen sonst nicht richtig interpretiert werden können und der Benutzer keinen Nutzen aus dem mobilen Arbeiten ziehen kann.

3.1.3. Benutzerabhängig

Da jeder Mensch anders ist, hat er natürlich auch andere, im Sinne von eigene, Eindrücke. Es gibt Menschen, die sich trotz einer hohen Anzahl an Elementen ein kleines Modell gut vorstellen können und es gibt Menschen, die auf einem kleinen Display bereits Probleme mit wenigen Elementen haben. Daher muss man bei der Applikation auf den Benutzer Rücksicht nehmen. Es wäre sicherlich von Vorteil, wenn der Benutzer unabhängig von den oben genannten Argumenten und Erkenntnissen selbst bestimmen könnte, ob das Modell textuell oder graphisch dargestellt werden soll. Es kann auch die Möglichkeit geben, dass man nur Teilausschnitte des Modells anzeigt. Dies könnte für manche Benutzer sehr hilfreich sein.

3.2. Beispiele für die Anpassung von Modellnotationen

In diesem Kapitel werden unterschiedliche Modellierungssprachen als Beispiele angeführt. Dabei werden die Notationen an die bereits genannten Kriterien angepasst und umgeformt. Folgende Anpassungen werden durchgeführt:

- Graphische Aufbereitung: Es wird nur die „oberste Ebene“ des Modells angezeigt und der Benutzer entscheidet selbst, bei welchem Abschnitt des Modells weitere Details angezeigt werden sollen.
- Zentrierte Ansicht: Der Benutzer kann auswählen, aus welcher Perspektive er ein Modell sehen möchte, und kann sich das Modell aus dem Blickwinkel von einzelnen Elementen anzeigen lassen.
- Textuell: Die Modellnotation wird nicht mehr mit Hilfe von Elementen gezeichnet, sondern mit Textbausteinen umschrieben.
- Farblich: Mit Hilfe von Farben können Textbausteine noch besser voneinander abgegrenzt werden und der Text gestaltet sich übersichtlicher für den Benutzer.
- Reduktion von Informationen: Unwichtige Informationen werden zum besseren Verständnis des Modells ausgeblendet und erst durch Interaktion des Benutzers geladen und angezeigt.

3.2.1. Klassendiagramm

Ein Klassendiagramm kann aus einer oder mehreren Klassen bestehen und eine Klasse wird als Rechteck dargestellt. Dieses Rechteck ist in drei Teile unterteilt:

- Notation Teil 1: Schlüsselwort, Stereotyp, Name der Klasse und Eigenschaft
- Notation Teil 2: Attribute der Klasse
- Notation Teil 3: Operationen der Klasse

Eine Klasse muss nicht zwingend aus drei Teilen bestehen, falls die Klasse keine Attribute oder Operationen enthält, können diese Teile ausgelassen werden. In der Abbildung 10 wird ein Beispiel für eine Klasse dargestellt:

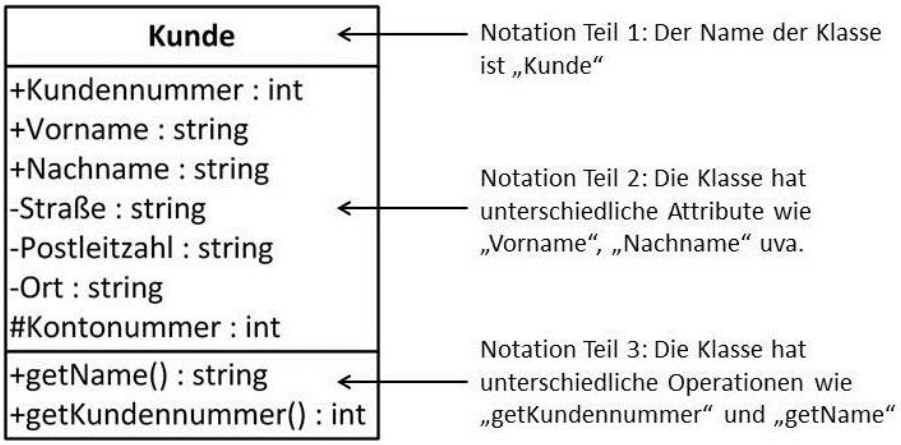


Abbildung 10 zeigt eine Klasse mit dem Klassennamen Kunden und unterschiedlichen Attributen und Operationen.

Die Klasse soll einen Kunden darstellen, der unterschiedliche Attribute hat, wie Vorname, Nachname, Kundennummer, Straße, Postleitzahl, Ort und Kontonummer. Die Klasse stellt zwei unterschiedliche Operationen zur Verfügung: getName, diese soll den Namen des Kunden und getKundennummer, diese soll die Kundennummer des Kunden zurückgeben.

3.2.1.1. Notation Teil 1: Schlüsselwort, Stereotyp, Name der Klasse und Eigenschaft

Man kann in der Abbildung 10 den Namen der Klasse erkennen. Das Schlüsselwort oder der Stereotyp wird immer in „<< >>“ Klammern über dem Klassennamen geschrieben. In der UML gibt es fast 100[11] verschiedene Schlüsselwörter und beinahe 40[12] davon repräsentieren auch einen Stereotyp. Der Stereotyp definiert zusätzliche Metaattribute und Einschränkungen, die die Klasse weiter spezialisieren können. Der nächste Begriff stellt den Namen der Klasse dar. Der unterste Begriff stellt die Eigenschaft der Klasse

[11] Eine vollständige Auflistung aller Schlüsselworte kann man unter anderem in der Quelle (UML@Work S. 378 f) nachlesen.
[12] Eine vollständige Auflistung aller Stereotypen kann man unter anderem in der Quelle (UML@Work S. 381 f) nachlesen.

dar und wird immer in „{ }“ Klammern geschrieben. Hier könnte zum Beispiel der Ausdruck „abstract[13]“ angeführt werden.

3.2.1.2. Notation Teil 2: Attribute der Klasse

Eine Klasse kann unterschiedliche Attribute enthalten, die jeweils unterschiedliche Sichtbarkeiten und Datentypen enthalten können. Wie man in der Abbildung 10 im Bereich des Teils 2 sehen kann, wurden verschiedene Attribute mit unterschiedlichen Datentypen hinzugefügt. Die Sichtbarkeit der Attribute wird mit den folgenden vier Symbolen dargestellt:

- „+“ (public): Der Zugriff ist für alle möglich
- „#“ (protected): Der Zugriff ist nur innerhalb der Klasse und von Unterklassen möglich
- „-“ (private): Der Zugriff ist nur innerhalb der Klasse möglich
- „~“ (package): Der Zugriff ist nur innerhalb des Pakets möglich

3.2.1.3. Notation Teil 3: Operationen der Klasse

Operationen werden innerhalb der Klasse programmiert, die beispielsweise die Inhalte von Attributen zurückliefern oder unterschiedliche Berechnungen durchführen können. Eine Operation kann, so wie auch Attribute, unterschiedlich sichtbar für andere Klassen sein. Bei Operationen werden zusätzlich noch Angaben über Variablen, die zur Ausführung benötigt werden, gemacht und auch Angaben über einen eventuellen Rückgabewert werden dort hinterlegt. In der Abbildung 10 können die zwei Operationen in Teil 3 gefunden werden. Die Operationen sind beide public und eine soll den Namen des Kunden (getName), die andere hingegen die Kundennummer (getKundennummer) zurückliefern. Die Datentypen hinter dem Doppelpunkt stellen den Datentyp des Rückgabewerts dar.

Falls ein Klassendiagramm aus mehr als einer Klasse besteht, was meistens der Fall sein wird, werden die Klassen in Beziehungen zueinander stehen. Diese Beziehungen können unterschiedlicher Art sein:

[13] Wenn eine Klasse in der Java-Programmierung „abstract“ ist, heißt das, diese Klasse kann nicht initialisiert werden, jedoch kann man Subklassen ableiten. Eine „abstract“ Methode besitzt keine Implementierung und sobald eine Methode „abstract“ ist, muss auch die Klasse „abstract“ sein. (Abstract Java Code)

- Generalisierung (auch Vererbung genannt): Die Generalisierung im Klassendiagramm stellt eine „Generelle-Klasse“ und eine „Spezifische-Klasse“ dar. Die „Spezifische-Klasse“ erbt im Normalfall alle Attribute und Methoden der „Generellen-Klasse“. Man könnte hier auch Redefinitionen in der „Spezifischen-Klasse“ durchführen. In Abbildung 11 sieht man die zwei Klassen „Kunde“ und „Premium-Kunde“. Der Premium-Kunde erbt die essentiellen Attribute wie Nachname und Vorname, hat jedoch auch eigene definierte Attribute wie den verbesserten Zinssatz oder auch die Stufe, in die der Premium-Kunde eingeordnet wurde. Es können auch mehrere unterschiedliche Klassen von einer Klasse erben, jedoch ist auf direktem Weg, zumindest bei der Programmiersprache Java, nur die Einfachvererbung möglich. (UML@Work) (Java Insel 5)

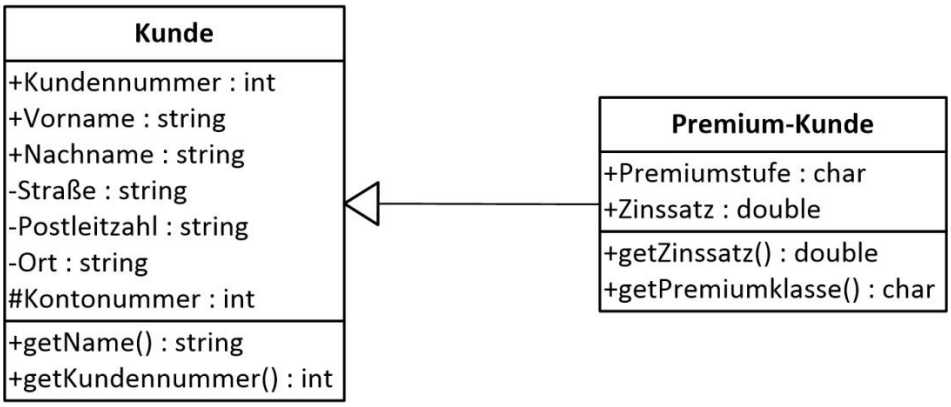


Abbildung 11 zeigt 2 Klassen, in welcher die „Premium-Kunde-Klasse“ von der „Kunde-Klasse“ erbt.

- Assoziationsgeneralisierung: Man kann zwei Klassen miteinander verbinden und festlegen, wie diese Klassen miteinander „interagieren“. In der Abbildung 12 kann man sehen, dass ein Kunde zwischen 0 und beliebig viele Sparbücher haben kann. Es können auch mehrere Klassen miteinander interagieren. (UML@Work)

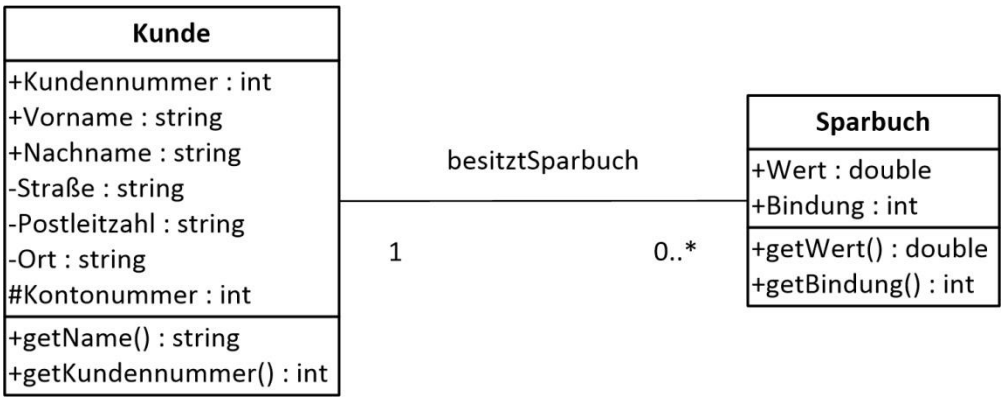


Abbildung 12 zeigt 2 Klassen, in welcher die „Sparbuch-Klasse“ mit der „Kunde-Klasse“ interagiert. Ein Kunde kann keines, eines oder mehrere Sparbücher besitzen.

- Verwendung: Mit der Verwendungsbeziehung wird gezeigt, welche Klassen miteinander interagieren bzw. welche Klasse von welcher Klasse verwendet wird. In der Abbildung 13 wird gezeigt, dass ein „Kundentermin“ die Klasse „Kunde“ und die Klasse „Berater“ verwendet. In diesem Fall kann man sich vorstellen, dass der Kundentermin ohne Kunde und ohne Berater nicht viel Sinn ergibt und daher diese Klasse die anderen beiden Klassen verwenden sollte. (UML@Work)

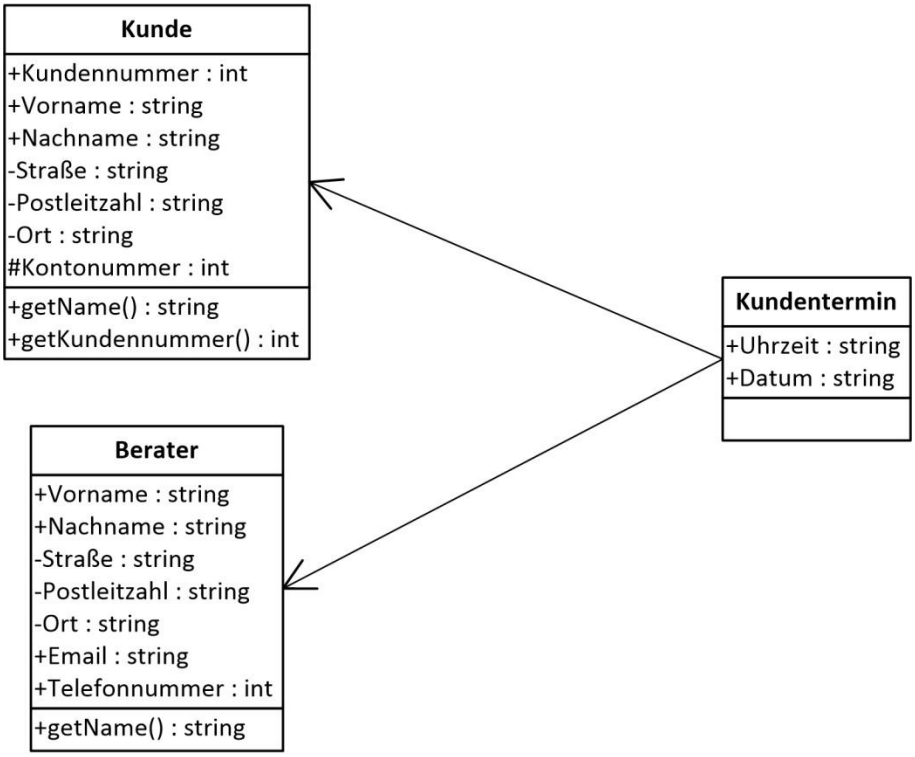


Abbildung 13 zeigt 3 Klassen, in welcher die „Kundentermin-Klasse“ die Klassen „Kunde“ und „Berater“ verwendet.

- Komposition und Aggregation: Diese beiden Beziehungen werden mit Hilfe einer ausgefüllten Raute (Komposition) bzw. einer leeren Raute (Aggregation) dargestellt. Die Komposition entspricht immer einer 0…1 Beziehung und stellt eine strengere Form der Aggregation dar. Die Aggregation kann eine 0…*:0…* Beziehung widergeben. Die Aggregation wird als die „Besteht-aus-Beziehung“ oder auch als Meister-Sklave-Verbindung bezeichnet. Die Raute ist immer beim „stärkeren“ Ende der Beziehung angesiedelt. Die Komposition stellt die Verbindung zweier Elemente dar, wobei eines nicht ohne den anderen existieren kann. In der Quelle (UML@Work) wird als Beispiel ein Formular im Internet herangezogen. Alle Eingabefelder sind mit Eingabeformular verbunden, jedoch kann keines der Eingabefelder existieren ohne das große komplette Formular. (UML@Work) (Java Insel 5)

3.2.1.4. Adaption

Klassendiagramme können bei größeren Softwareprojekten sehr große Dimensionen annehmen. Softwareprojekte mit 50-100 Klassen sind keine Seltenheit und gehören durchaus zum Alltag eines jeden Programmierers. Diese Diagramme können sehr schnell unübersichtlich werden. Auch auf einem „normalem“ Bildschirm wird die Orientierung im Diagramm immer schwieriger, daher muss bei der Darstellung auf dem Display eines Smartphones dem Benutzer geholfen werden, damit es möglich ist, sich in dem Diagramm zurechtzufinden. Wenn man alle vorangegangen Beispiele in einem Diagramm darstellt, sieht es wie folgt aus:

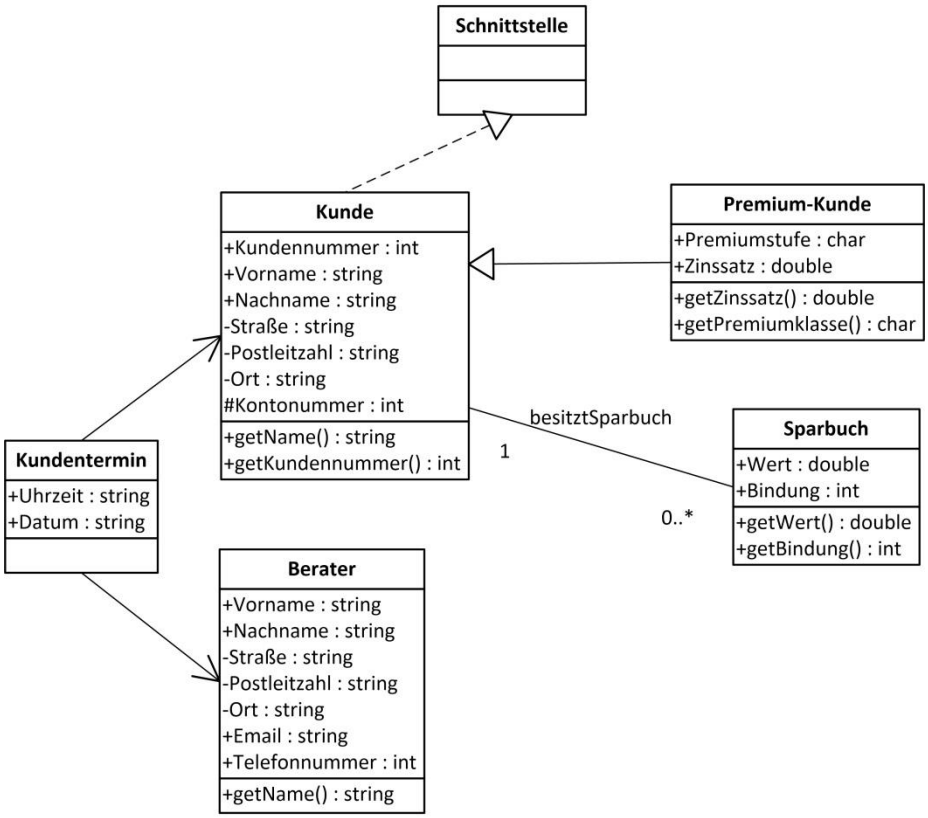


Abbildung 14 zeigt ein Klassendiagramm von typischen Geschäftsprozessen in einer Bank, wie etwa, dass ein Kunde ein Sparbuch haben kann oder dass ein Kunde ein Premium-Kunde sein kann. Weiters kann es einen Kundentermin zwischen einem Kunden und einem Berater geben.

Es kann, wie bereits in der Einleitung des Kapitels 3.1 erwähnt, zwei unterschiedliche Möglichkeiten geben, um diese Darstellung für den Endbenutzer zu vereinfachen:

- **Textuell**

Die textuelle Darstellung ist im Falle des Klassendiagramms eine durchaus plausible Möglichkeit. Das Klassendiagramm selbst, wie in Abbildung 14 gezeigt, besitzt schon sehr viel Text und besteht aus fast immer denselben Elementen. Die Elemente können zwar ein wenig unterschiedlich sein, wie etwa ein Interface, eine Klasse oder auch eine Schnittstelle, aber das Element ist im Grunde immer dasselbe. Die Beziehungen können ganz unterschiedlich sein und werden immer unterschiedlich dargestellt, wie etwa die Art des Pfeiles (voll, leer, offen oder geschlossen) oder auch mittels einer gezogenen oder gestrichelten Linie.

Es gibt aber auch die Möglichkeit das Modell textuell darzustellen und jede Klasse mit allen Attributen, Sichtbarkeiten und Datentypen aufzuschreiben. Der Übersichtlichkeit nach sollte man dies als Aufzählung machen und da den meisten Benutzern die Notation des Klassendiagramms geläufig ist, kann man Datentypen und Sichtbarkeit abkürzen. Daraus folgt, dass man jede Klasse in textueller Form ähnlich wie in graphischer Form aufbereiten würde. Die unterschiedlichen Beziehungen kann man textuell einfach mit Hilfe von Bausteinen niederschreiben:

Beispiel 1:

Die Klasse Kunde besitzt die Klasse Schnittstelle. Die Klasse Premium-Kunde erbt von der Klasse Kunde. Die Klasse Kundentermin verwendet die Klasse Kunde und verwendet die Klasse Berater. Die Klasse Kunde (1) hat eine Assoziationsgeneralisierung (besitztSparbuch) mit der Klasse Sparbuch (0…*)

Das Hinzufügen von Attributen würde das Ganze sehr schlecht lesbar machen:

Beispiel 2:

Die Klasse Premium-Kunde [(public,Premiumstufe,character) (public,Zinssatz,double) | (public,getZinssatz(),double) (public,getPremiumklasse(),character)] erbt von der

Klasse Kunde [(public,Kundennummer,integer) (public,Vorname,string) (public,Nachname,string) (private,Straße,string) (private,Postleitzahl,string) (private,Ort,string) (protected,Kontonummer,integer) | (public,getName(),string) (public,getKundennummer(),integer)].

So könnte man textuell einen allgemeinen Überblick der Klassen und ihrer Beziehungen präsentieren (siehe Beispiel 1) und dem Benutzer die Möglichkeit geben, per Link auf eine Klasse mit dem Finger zu klicken, um dann die Attribute und Operationen der Klasse aufgelistet zu bekommen. Dies könnte in einem Pop-up passieren, das man einfach wieder schließen kann und zum Überblick der textuellen Beschreibung zwischen Beziehungen und Klassen zurückkehrt. Sobald sich das Pop-Up öffnet, werden Daten „on-demand“ nachgeladen. Dieses Verfahren hat den Vorteil, dass nur notwendige Information geladen werden und erst wenn der Benutzer detaillierte Informationen benötigt, werden diese auch geladen. In der Informatik spricht man von einem Designpattern namens „Lazy Loading“ (übersetzt aus dem Englischen: faules Laden). Bei diesem Pattern werden Informationen zwar zur Verfügung gestellt, aber erst bei explizitem Nachfragen auch angezeigt. Ein Beispiel dafür kann man in folgender Abbildung sehen:

Abbildung 15 zeigt ein textuelle Beschreibung der Beziehungen eines Klassendiagramms mit der Möglichkeit sich Klassen mithilfe eines Pop-ups detailliert anzeigen zu lassen. (Abbildung des Smartphones Zitat (HTC Magic))

- **Graphisch**

Wie bereits im letzten Abschnitt erwähnt, ist im Klassendiagramm bereits sehr viel textuell erfasst. Die Attribute und Operationen einer Klasse werden bereits textuell im Modell niedergeschrieben und vermerkt. In den Klassen werden einige Abkürzungen für Datentypen verwendet und Sichtbarkeiten mit unterschiedlichen Symbolen dargestellt. Auch für die graphische Darstellung gilt, dass Klassendiagramme sehr groß ausfallen können und dass man hier möglicherweise auf einem kleinen Device Probleme haben kann, alles zu erkennen. Das „Big picture“ kann hier nur sehr schwer vom menschlichen Auge erfasst werden und man muss bei mehr als 10 Klassen bereits sehr viel am Touchscreen vergrößern und scrollen, um alles zu sehen. Auch bei der graphischen Darstellung sollte man wieder in 2 Ebenen arbeiten. Die erste Ebene soll die Beziehungen zwischen den Klassen darstellen und die zweite Ebene detailliert die

Inhalte der Klasse mit Hilfe eines Pop-ups widergeben. In der folgenden Abbildung sind Beispiele dafür zu erkennen:

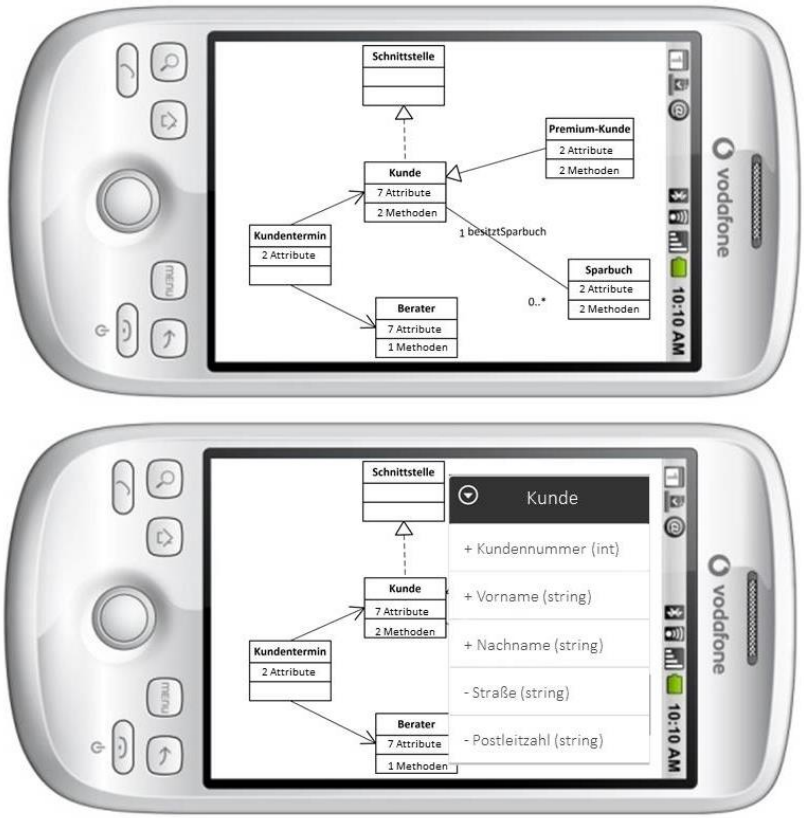


Abbildung 16: 2 Smartphones befinden sich in horizontaler Lage. Das obere Smartphone zeigt die erste Ebene des Diagramms. Das untere Smartphone stellt die zweite Ebene, die Detailsicht auf die Klasse Kunde, dar. (Abbildung des Smartphones Zitat (HTC Magic))

Man kann in der Abbildung 16 sehen, dass die Übersicht über alle Klassen und deren Beziehungen gut erkennbar sind. Alle Klassen haben den Namen der Klasse eingetragen und im zweiten Abschnitt des Elements die Anzahl der Attribute stehen. Im dritten Abschnitt des Elements wird die Anzahl der Operationen angezeigt. Die detaillierte Sicht wird im unteren Smartphone dargestellt und man kann in dem Pop-up erkennen, welche Attribute und Methoden die Klasse besitzt. Die Übersichtlichkeit ist sehr gut gegeben und man kann so auch größere Diagramme noch gut auf einem Smartphone darstellen.

Falls ein Modell zu groß wird (mehr als 20 Klassen), kann der Benutzer eine Aufzählung mit allen Klassen aufgelistet bekommen, bei der er eine einzelne Klasse auswählt und

nur Klassen dargestellt werden, die mit dieser Klasse in Beziehung stehen bzw. interagieren. Parallel dazu kann der Benutzer sich auch das ganze Diagramm anzeigen lassen und mit der Klassen-zentrierten-Sicht sich einzelne Ausschnitte des Modells genauer ansehen.

Bei sehr großen Modellen kann man auch Farben zur besseren Übersichtlichkeit einsetzen. Zusammenhängende Klassen können mit farbigen Beziehungen dargestellt werden. Die Variablen können auch farbig dargestellt werden. In so einem Fall könnten public Variablen rot, private Variablen grün und protected Variablen blau dargestellt werden. Damit kann man die Zeichen vor den Variablen reduzieren. Weiters kann man mit kursiver, fetter oder unterstrichener Schrift die gängigsten Datentypen darstellen und spart sich damit die Bezeichnung hinter der Variablen.

- **Vergleich**

Die Anpassungen der Modellnotationen können wie folgt aussehen:

- Graphische Aufbereitung: Im Fall des Klassendiagramms wird die graphische Aufbereitung durch Reduktion von Informationen oder eine zentrierte Ansicht durchgeführt.
- Zentrierte Ansicht: Eine Klasse wird ausgewählt und nur Klassen, die mit der ausgewählten Klassen direkt verbunden sind, werden angezeigt.
- Textuell: Das Modell wird textuell mit Hilfe von Textbausteinen beschrieben. Diese Art der Notationsveränderung kann mit farblichen Mitteln oder mit der zentrierten Sicht kombiniert werden. Es können Standardbeziehungen farblich angezeigt werden. In der zentrierten Sicht wählt der Benutzer wieder eine Klasse aus, die dann textuell mit allen Beziehungen dargestellt wird.
- Farblich: Um eine bessere Übersicht über ein Modell zu bekommen, kann man Klassen farblich gestalten und verbinden. Es wäre auch möglich die zentrierte Sicht mit Farben zu kombinieren. Es wird eine Klasse ausgewählt und alle direkt verbunden Klassen werden grün angezeigt, alle erbenden Klassen rot und alle Klassen, die an die ausgewählte Klassen vererben, blau angezeigt.
- Reduktion von Informationen: Die Reduktion von Informationen kann wie in Abbildung 16 passieren, in dem Attribute und Methoden erst auf Benutzerwunsch nachgeladen werden. Die Reduktion der Informationen kann

man auch mit der zentrierten Ansicht kombinieren, indem man nur eine zentrierte Klasse und die direkt verbunden Klassen anzeigt. Mit einem „+“ und einem „-“ Knopf können on demand mehr Klassen nachgeladen werden. Mit Hilfe des „+“-Knopfs können Klassen, die über eine andere Klasse mit der ausgewählten Klasse verbunden sind, nachgeladen werden und mit dem „-“-Knopf verschwindet diese „Ebene“ wieder.

In den zwei vorangegangenen Abschnitten wurde bereits auf den Vorteil und zugleich auch auf die Notwendigkeit der Aufteilung des Diagramms in zwei Ebenen hingewiesen, egal ob die Darstellung textuell oder graphisch passieren soll. Weiters kann man im Abschnitt der textuellen Darstellung auf der Abbildung 16 erkennen, dass bereits ein kleines Modell mit nur 6 Klassen und 5 Beziehungen den gesamten Bildschirm in textueller Beschreibung einnimmt. Wenn ein Modell 20 oder mehr Klassen und dementsprechend viele Beziehungen aufweist, kann es in der textuellen Darstellung sehr schnell sehr unübersichtlich werden. Hier liegt der Vorteil eindeutig in der graphischen Darstellung, die in der adaptierten Form auch größere Klassendiagramme zulässt. Dabei unterstützt die Klassen-zentrierte-Sicht den Benutzer, Teile aus großen Modellen auszuwählen und im Detail zu betrachten. In folgender Abbildung kann man den direkten Vergleich ansehen:

Abbildung 17 das Smartphone links zeigt das Originalmodell, das Smartphone in der Mitte zeigt den Text und das Smartphone rechts zeigt das adaptierte Modell (Abbildung des Smartphones Zitat (HTC Magic))

Man kann in der Abbildung 17 eindeutig erkennen, dass das adaptierte Modell die beste Übersicht bietet. Durch „Lazy Loading“ kann man die noch unsichtbaren Informationen mit nur einem Klick nachladen.

Durch die Zoomfunktion der Smartphones ist es möglich, sich zuerst das Modell im Ganzen und dann einzelne Klassen und deren Abhängigkeiten und Beziehungen anzusehen. In der textuellen Form wäre dies zwar auch möglich, aber die Darstellung und der Ausschnitt aus dem Modell würde nicht den gewünschten Effekt bringen.

3.2.2. Business Process Model and Notation (BPMN)

Die „Business Process Model and Notation“ (übersetzt aus dem Englischen: Geschäftsprozessmodellnotation) (OMG BPMN) wird zur Darstellung von Geschäftsprozessen und auch Workflows (übersetzt aus dem Englischen: Arbeitsablauf) verwendet. Ein Geschäftsprozessmodell kann aus vielen verschiedenen Elementen bestehen und daher sehr groß werden. Da das Business Process Modell von links nach

rechts modelliert bzw. gelesen wird, benötigt man dementsprechend viel Platz um die Abbildung darzustellen. Auf kleinen Devices soll das Modell etwas verkleinert werden, um es lesbar darstellen zu können. Diese Anpassungen werden im Kapitel 3.2.2.2 (Adaption) genauer vorgestellt.

3.2.2.1. Notation

Das Modell besteht aus einem oder mehreren Pools, die wiederum aus einer oder mehreren Schwimmbahnen (englisch: Swimlanes) bestehen. Dies wird in folgender Abbildung dargestellt:

Abbildung 18 zeigt einen Pool mit dem Namen „Bank“, der zwei Schwimmbahnen mit den Namen „Kunde“ und „Mitarbeiter“ enthält.

Ein Pool stellt eine Benutzerrolle (einen Benutzer selbst) oder ein System dar. In der Abbildung 18 *ist ein Pool („Bank“) mit unterschiedlichen Schwimmbahnen („Mitarbeiter“ bzw. „Kunde“)* zu erkennen. Sowohl der Pool, als auch die Schwimmbahnen können miteinander interagieren.

Schwimmbahnen können unterschiedliche Elemente enthalten:

- Aktivitäten (english: Activities): werden als „Task“ (Aufgabe) dargestellt und können aber auch als Subprozess mehrere Aktivitäten enthalten. In folgender Abbildung werden die zwei unterschiedlichen Aktivitäten dargestellt.

Abbildung 19 zeigt auf der linken Seite eine Aktivität mit dem Namen „Bargeldeinzahlung“ und auf der rechten Seite einen Subprozess namens „Sparbucheinzahlung“

- Entscheidungspunkte (english: Gateways): können den Weg des „Flusses“ in unterschiedliche Richtungen lenken. Beim Entscheidungspunkt können mehrere „Flüsse“ zusammentreffen oder auch mehrere Flüsse entstehen. Das Symbol im Inneren des Entscheidungspunktes steht für: „und“ (english: AND), „oder“ (english: OR), „xoder“ (english: XOR) oder für Ereignisbasiert. Diese unterschiedlichen Symbole werden in Abbildung 20 dargestellt.

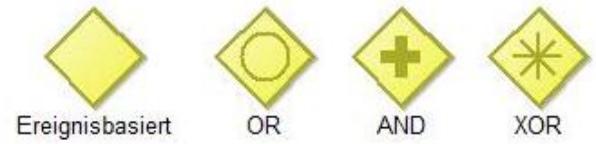


Abbildung 20 zeigt die unterschiedlichen Entscheidungspunkte mit den zutreffenden Namen unter dem Entscheidungsviereck.

Der „AND-Gateway“ besagt, dass alle Wege weiter verfolgt werden müssen bzw. dass man auf alle Wege warten muss, bis diese am Entscheidungspunkt eingetroffen sind. Der „OR-Gateway“ besagt, dass mindestens ein Weg weiter verfolgt werden muss bzw. sobald ein Fluss am Entscheidungspunkt angekommen ist, wird dieser weiterverarbeitet und der Rest wird ignoriert. Durch den „XOR-Gateway“ wird nur ein Fluss weiter verfolgt. Der Fluss der Entscheidungsbasierte Gateway wird durch das zeitlich erste Eintreten eines Events ermittelt.

- Ereignisse (english: Events): werden mit einem Kreis (wie in Abbildung 21 ersichtlich) dargestellt und können in einem Geschäftsprozessmodell entweder

als "Catch"[14] (deutsch: fangen) oder als "Throw"[15] (deutsch: werfen) Event auftreten. Es gibt 3 unterschiedliche Ereignisklassen:

- Start (deutsch: Start): Ein Ereignis wird gestartet
- Intermediate (deutsch: Zwischen): Ein "Zwischenereignis", das entweder während eines normalen "Flows" auftaucht oder das zu einer Aktivität hinzugefügt wird.
- End (deutsch: Ende): Ein Ereignis wird beendet

Ein Event kann von unterschiedlichem Typus sein[16]. Es gibt Nachrichten, Timer, Fehler, Abbruch, Signal und einige andere Eventtypen in der BPMN. Je nachdem welcher Eventtyp verwendet wird, bekommt der Kreis ein anderes Symbol in der Mitte. Für ein Nachrichten-Event wird ein Brief eingesetzt oder für ein Timer-Event eine Uhr. Folgende Abbildung zeigt die unterschiedlichen Möglichkeiten von Ereignissen:

Abbildung 21 zeigt die unterschiedlichen Ereignisformen, die bei der BPMN eingesetzt werden können

Diese angeführten Elemente werden mit drei unterschiedlichen „Flows“ verbunden:

[14] Ein sogenanntes Catch-Event kann ein „Start“ Event oder ein „receive“ (deutsch: empfangen) Event sein.
[15] Ein sogenanntes Throw-Event kann ein „End“ Event oder ein „send“ (deutsch: senden) Event sein.
[16] siehe Quelle (OMG BPMN) S. 261

- Sequence Flow: Ist eine Verbindung zwischen zwei Elementen im Modell und stellt einen „gerichteten Graph“ zwischen diesen dar.
- Conditional Flow: Ist eine Verbindung zwischen zwei Elementen, die eine Bedingung erfüllen muss. Dieser Graph ist auch gerichtet und benötigt immer eine Alternative (Default Flow), falls die Bedingung nicht erfüllt wurde.
- Default Flow: Ist eine Alternative, die vom Conditional Flow benötigt wird, und kann auch bei Gateways zum Einsatz kommen.

Mit Hilfe der Elemente im Kapitel Notation kann man ein Geschäftsprozessmodell erstellen. In folgender Abbildung wird ein Modell gezeigt, das einen Geschäftsprozess zwischen einem Kunden und einem Kundenbetreuer in einer Bank darstellt. Der Kunde möchte Geld auf sein Sparbuch einzahlen und somit wird eine Interaktion zwischen den beiden Personen gestartet:

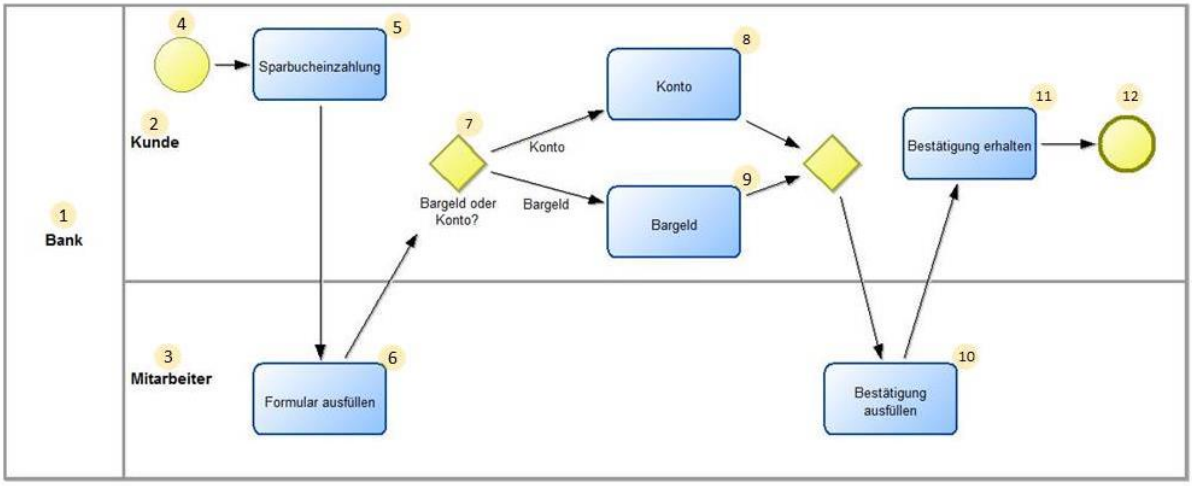


Abbildung 22 zeigt ein Geschäftsprozessmodell in einer Bank mit dem Ziel für den Kunden, dass eine Einzahlung auf sein Sparbuch getätigt werden soll. Der Flow geht hier von links nach rechts und es gibt unterschiedliche Interaktionen zwischen den beiden Personen „Kunde“ und „Mitarbeiter“.

In der Abbildung werden unterschiedliche Aktivitäten von unterschiedlichen Benutzern ausgeführt. Der Geschäftsprozess beginnt beim Kunden mit der Aktivität „Sparbucheinzahlung“ Danach geht die Ablauffolge weiter zum Mitarbeiter mit der Aktivität „Formular ausfüllen“. Auf diese Aktivität folgt „Bargeld oder Konto“, eine Entscheidung zwischen der Bargeldeinzahlung und der Überweisung von einem Konto.

Nachdem diese Entscheidung vom Kunden getroffen wurde, wird die Aktivität „Bestätigung ausfüllen“ vom Mitarbeiter ausgeführt und der Kunde muss die „Bestätigung entgegennehmen“. Danach endet der Geschäftsprozess. Dieses Modell ist ein relativ kurzes und einfaches Geschäftsprozessmodell ohne Subprozesse, Events oder komplexe Entscheidungen. Bei Geschäftsprozessen mit mehr Interaktionen oder Abläufen können diese Modelle durchaus sehr rasch sehr viele Elemente enthalten.

3.2.2.2. Adaption

Wie man in der Abbildung 22 sehen kann, ist das Geschäftsprozessmodell eher klein, aber trotzdem sollte man den Flow zwischen Aktivitäten, Subprozessen und Events eindeutig nachvollziehen können, um das Modell bestmöglich interpretieren zu können. Ein relativ kleines Modell, mit nur zehn Elementen, einem Pool und zwei Schwimmbahnen, hat schon kaum Platz auf einer Hochformat A4 Seite. Die Darstellung eines solchen Geschäftsprozessmodells auf einem Smartphone-Display kann durchaus zur Herausforderung werden. Das Beispiel aus Abbildung 22 (siehe Kapitel 3.2.2.1) wird in den folgenden zwei Unterpunkten in textueller und graphischer Form dargestellt. Im Anschluss wird nach einer optimalen Anzeige der Modellierungssprache BPMN auf Smartphones gesucht.

- **Textuell**

Eine textuelle Beschreibung des oben genannten Beispiels sollte mit Textbausteinen durchaus möglich sein. Der Vorteil hier wäre, dass man durch einen automatisierten Algorithmus die Elemente in Texte übersetzen lassen könnte und so einen Überblick bekommt, was in diesem Geschäftsprozess abläuft. Um die Übersichtlichkeit in so einem Fall zu ermöglichen, könnte man auch hier eine Auswahl treffen, um den Geschäftsprozess von einem Pool aus oder einer Schwimmbahn aus zu sehen. Es wäre von Vorteil, wenn man den Benutzer nicht mit dem ganzen Modell überfällt, sondern einzelne Zeilen herausfiltert und diese textlich erklärt. Der Benutzer kann so einen Überblick bekommen und sich das Modell vorstellen. In der textuellen Beschreibung kann leicht hinterlegt werden, wenn der Flow sich in eine andere Schwimmbahn bewegt und wenn er wieder in die ausgewählte Schwimmbahn zurückkehrt. Durch eine Nummer hinter der Auflistung der Schwimmbahnen kann man dem Benutzer zeigen, wie oft der Flow die Schwimmbahn verlässt. Damit kann der Benutzer selbständig

entscheiden, ob es überhaupt Sinn macht, bei der Anzeige andere Schwimmbahnen auszublenden oder ob dies sonst zu unübersichtlich wird. Der Text für das Beispiel aus Kapitel 3.2.2.1 würde wie folgt aussehen:

Pool: Bank[1] mit Schwimmbahn: Kunde[2] und Schwimmbahn: Mitarbeiter[3]
Schwimmbahn: Kunde[2] hat den Event „Start“[4]. Der Flow führt zu „Sparbucheinzahlung“[5]. Der Flow führt zu „Formular auswählen“[6] in der Schwimmbahn: Mitarbeiter.[3] Der Flow führt zum Entscheidungspunkt „Bargeld oder Konto?“[7] in der Schwimmbahn: Kunde.[2] Der Entscheidungspunkt führt zu 2 Aktivitäten: Konto, Bargeld.[8][9] Der Flow führt zu „Bestätigung ausfüllen“[10] in der Schwimmbahn: Mitarbeiter[3] Der Flow führt zu „Bestätigung erhalten“[11] in der Schwimmbahn: Kunde.[2] Der Flow führt zum Event „Ende“[12]

Dieser Text kann leicht auf einem Smartphone dargestellt werden und man kann die Textbausteine mit den Nummern in Abbildung 22 vergleichen. Mithilfe eines Algorithmus werden die Elemente der Abbildung in Textbausteine umgewandelt.

Abbildung 23 zeigt zwei Smartphones mit einer Auswahlliste auf der linken Seite und dem angezeigt Text der Schwimmbahn auf der rechten Seite. (Abbildung des Smartphones Zitat (HTC Magic))

Das linke der beiden Smartphones in Abbildung 23 zeigt eine Auswahlliste aller verfügbaren Schwimmbahnen in einem Pool. In diesem Fall geht es wieder um das Beispiel der Bank mit den zwei Schwimmbahnen „Kunde“ und „Mitarbeiter“. Das Smartphone auf der rechten Seite zeigt den Text der Schwimmbahn „Kunde“.

Durch den Flow, der im Geschäftsprozessmodell herrscht, kann man das Modell sehr leicht in Textbausteine umbauen und vor allem die Reihenfolge kann dem Benutzer sehr leicht dargestellt werden. Das Modell wird von links nach rechts betrachtet und der Text wird von oben nach unten gelesen. Das Geschäftsprozessmodell kann somit in der richtigen und eindeutigen Reihenfolge gelesen werden und die Übersichtlichkeit für den Benutzer ist mehr gegeben als beim Klassendiagramm oder beim Anwendungsfallmodell.

- **Graphisch**

Die Darstellung in graphischer Form wird bei den meisten Modellen in der BPMN sehr schwierig ausfallen, da die Modelle sehr groß werden können. Auch wenn man jedes moderne Smartphone nicht nur vertikal, sondern auch horizontal verwenden kann, wird ein Modell nicht übersichtlich genug dargestellt werden. Der Benutzer kann sich mit Hilfe der Multitouch-Technologie zwar Teile des Modells größer anzeigen lassen, es wird vermutlich bei den meisten Modellen aber trotzdem zu unübersichtlich sein, wie man in folgender Abbildung erkennen kann:

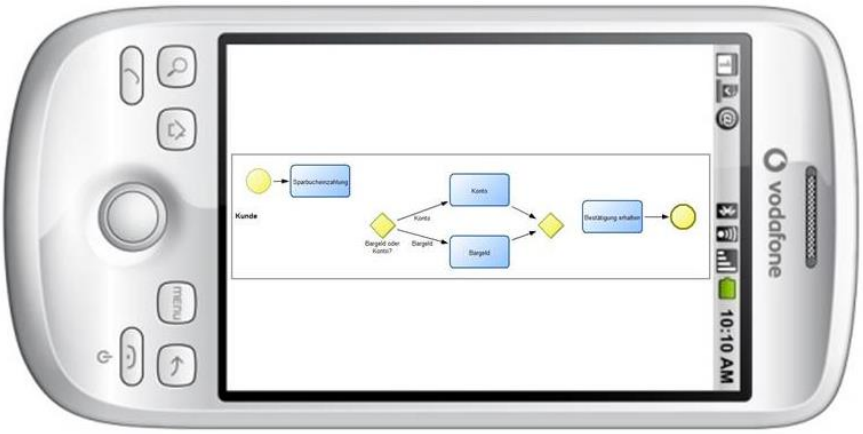


Abbildung 24 zeigt die Schwimmbahn „Kunde“ aus dem Beispiel „Bank“ (vgl.Abbildung 22) auf einem Smartphone (Abbildung des Smartphones Zitat (HTC Magic))

Um das Modell zu vereinfachen, könnte man es nur auf einen Pool oder eine Schwimmbahn reduzieren. Im Falle des Beispiels aus Abbildung 22 gibt es jedoch nicht mehr als einen Pool, auf den man das Modell reduzieren könnte. Auch der Flow springt zwei Mal zwischen den Schwimmbahnen hin und her, daher macht auch das Herausheben der einzelnen Schwimmbahnen keinen Sinn. Auch wenn man eine Schwimmbahn herausheben könnte, würde diese in den meisten Fällen noch immer sehr lange und auf einem kleinen horizontalen Display nicht gut zu erkennen sein. Dafür könnte der Benutzer das Modell mit Hilfe des Multitouchscreens näher heranzoomen und den Fokus auf einzelne Bereiche richten.

- **Vergleich**

Die Anpassungen der Modellnotationen können wie folgt aussehen:

- Graphische Aufbereitung: Ist für BPMN möglich. Man kann Subprozesse erst durch den expliziten Benutzerwunsch nachladen und anzeigen. Dadurch kann man das Modell zwar verkleinern, jedoch bleibt die Länge des Modells erhalten.
- Zentrierte Ansicht: Wie im vorherigen Abschnitt bereits erwähnt, kann der Benutzer sich einzelne Schwimmbahnen anzeigen lassen. Hier gehen leider sehr viele Informationen verloren und das Modell wird in den meisten Fällen sehr schwer interpretierbar sein.
- Textuell: Alle Elemente werden mit Hilfe von Textbausteinen angezeigt. Hier kann so wie in der graphischen Aufbereitung ein Subprozess mit einem „Link“ ausgestattet werden, um diesen nach Bedarf nachzuladen. Die Informationen werden zur besseren Übersicht in einem Pop-up Fenster dargestellt.
- Farblich: Texte zu Schwimmbahnen und / oder Pools können mit gleichen Farben gezeigt werden, um Zusammenhänge besser herauslesen zu können. Genauere Informationen zu Subprozessen, Konten oder Elementen, die erst auf speziellen Userwunsch gezeigt werden sollen, können mit einer eigenen Farbe (zum Beispiel blau) und unterstrichen dargestellt werden.
- Reduktion von Informationen: Informationen zu Subprozessen, Knoten oder Elementen werden erst nach ausdrücklichem Benutzerwunsch nachgeladen. Der Benutzer kann mit Hilfe eines Parameters festlegen, wie viele Objekte auf einmal angezeigt werden sollen. Dies kann Schwimmbahn oder Pool oder beides Übergreifend sein.

Wie man bereits durch die vielen unterschiedlichen Elemente in der Beschreibung erkennen kann, kann diese Art des Modells sehr groß und unübersichtlich werden. Bereits die Modellierung eines einfachen Geschäftsprozesses, wie etwa das Einzahlen eines Geldbetrags auf ein Sparbuch bei einer Bank (vgl. Abbildung 22), kann sehr groß ausfallen und auf einem Smartphone Screen sehr schlecht erkennbar sein. Wie bereits in der Einleitung des Kapitels 3.1.2 erwähnt, kann sich der Mensch nur durch langes Ansehen einer Szene mehr als 5 Objekten einprägen. Daher sollte man den Benutzer nicht mit zu vielen Objekten im Modell überfordern, vor allem da deren Darstellung auch sehr klein ausfallen würde. Subprozesse werden am besten durch anklickbare Links dargestellt, die Informationen „on-demand“ geladen. Der Subprozess im Subprozess verhält sich genauso. Der Benutzer muss sich in diesem Fall einfach durchklicken. Um das Modell etwas zu vereinfachen, kann der Benutzer über Parameter festlegen, wie viele Objekte in der obersten Ebene angezeigt werden sollen. Somit kann sich der Benutzer dann vorwärts und rückwärts durch das Modell hindurch klicken.

In der folgenden Abbildung kann man den Vergleich sehen zwischen der Modellierungsnation und den zwei adaptierten Notationen:

Abbildung 25 das linke Smartphone zeigt das Originalmodell, das mittlere Smartphone zeigt die zentrierte Sicht auf den Kunden in textueller Form und das rechte Smartphone zeigt die zentrierte Sicht auf den Kunden in graphischer Form. (Abbildung des Smartphones Zitat (HTC Magic))

In der Abbildung 25 kann man gut erkennen, dass die zentrierte Sicht bei einem so kleinen Modell graphisch nicht viel Verbesserung bringt. Das Problem bei der BPMN-Notation liegt nicht in der Breite des Modells sondern in der Länge.

Wie bereits im Kapitel Notation erwähnt, gibt es 3 unterschiedliche Eventklassen mit 13 unterschiedlichen Eventtypen. Alle diese Symbole auf einem 4" großen Display zu unterscheiden und nachvollziehen zu können, wie sich der Flow des Modells bewegt, ist eine sehr große Herausforderung. Ein bequemes und produktives mobiles Arbeiten wäre in so einem Fall nicht mehr gegeben. Dies würde für eine Umsetzung in textuelle Form sprechen. Man würde mit Hilfe von Textbausteinen und automatisch generierten Textbeschreibungen dem User eine bessere Möglichkeit bieten, das Modell auf einem Smartphone abzubilden.

3.2.3. Anwendungsfalldiagramm

Das „Use Case Diagram“ (Anwendungsfalldiagramm) ist eine der Modellierungssprachen, die in der UML definiert worden sind. Es soll, wie der Name schon sagt, Anwendungsfälle darstellen, mit deren Akteuren (Usern), Beziehungen und Abhängigkeiten. Ein Use Case Diagramm soll in der Informatik veranschaulichen, wie sich eine Anwendung in einem System verhalten kann bzw. verhalten wird. Im Anwendungsfalldiagramm werden Benutzer dargestellt, die unterschiedliche Anwendungsfälle ausüben können. Diese Anwendungsfälle können zu anderen Anwendungsfällen in einer Beziehung oder auch Abhängigkeit stehen. Das Anwendungsfalldiagramm besteht aus folgenden Elementen:

3.2.3.1. Benutzer (User)

Ein Benutzer, oder auch Akteur genannt, kann zu einer oder mehreren Anwendungen in Beziehung stehen. Der Benutzer wird wie in Abbildung 26 als „Strichmännchen“ dargestellt. Benutzer selbst können auch in Beziehung zu anderen Benutzern stehen. In dem vorliegenden Beispiel gibt es zwei verschiedene Benutzer, „Kunden“ und „Berater“, und beide „erben“ von dem Benutzer „Mensch“. Dies wird in folgender Abbildung 26 dargestellt. (UML@Work)

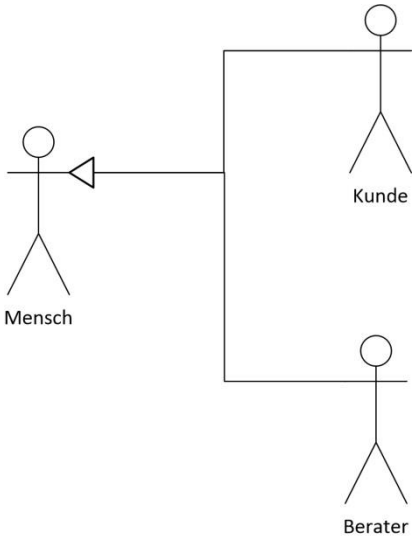


Abbildung 26 zeigt die Generalisierung eines Benutzers in einem Anwendungsfalldiagramm

3.2.3.2. Anwendungsfall (Case)

Ein Anwendungsfall ist ein „Ziel“, das der Benutzer erreichen möchte. Der Anwendungsfall wird nur kurz im Diagramm benannt. Er kann im klassischen Sinn gleich wie die Zielerreichung lauten oder er muss in einer externen Dokumentation näher beschrieben werden. Das Diagramm soll nur die Anwendung darstellen und in welcher Beziehung / Abhängigkeit diese zu anderen Elementen steht. (UML@Work)

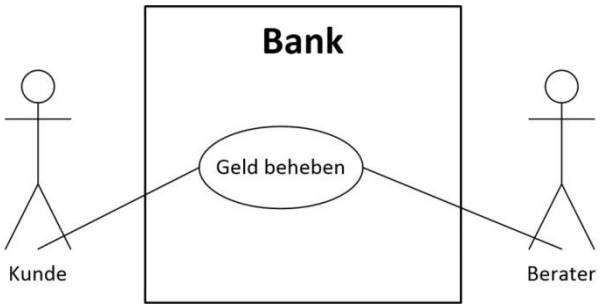


Abbildung 27 zeigt eine „Bank“, die einen Anwendungsfall namens „Geld beheben“ hat, sowie einen Benutzer-„Kunden“ und einen Benutzer-„Berater“, die diesen Anwendungsfall benutzen.

3.2.3.3. Beziehung

Die Beziehungen in dem Anwendungsfalldiagramm können zwischen Benutzern, zwischen Anwendungsfällen oder zwischen Anwendungsfällen und Benutzern auftreten. Man unterscheidet im Allgemeinen vier klassische Beziehungen zwischen den Elementen:

- Assoziation: siehe Kapitel 3.2.1.3. Man kann die Beziehung in Abbildung 27 sehen.
- Generalisierung: wie oben bereits in Kapitel 3.2.1.3 erklärt.

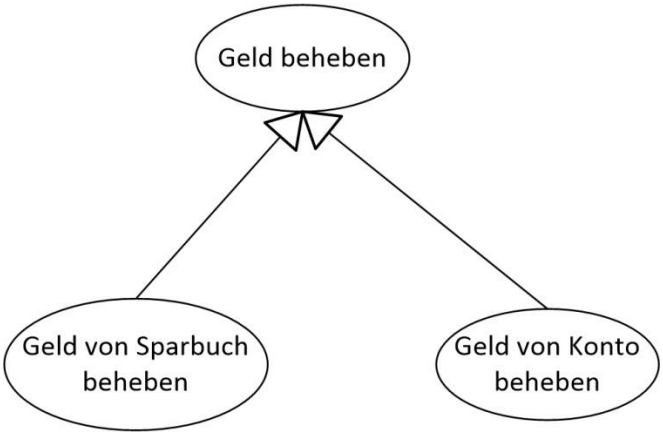


Abbildung 28 zeigt die Generalisierung von 2 speziellen Anwendungsfällen („Geld von Sparbuch beheben“ & „Geld von Konto beheben“) zu einem Fall („Geld beheben“)

- Include: die Include-Beziehung zeigt, dass ein Anwendungsfall einen anderen „inkludiert“. In dem folgenden Fall wird gezeigt, dass ein Anwendungsfall zwei weitere Fälle inkludiert. Man kann hier auch davon ausgehen, dass, falls man diesen Fall ausführt, auch die anderen beiden Fälle ausgeführt werden.

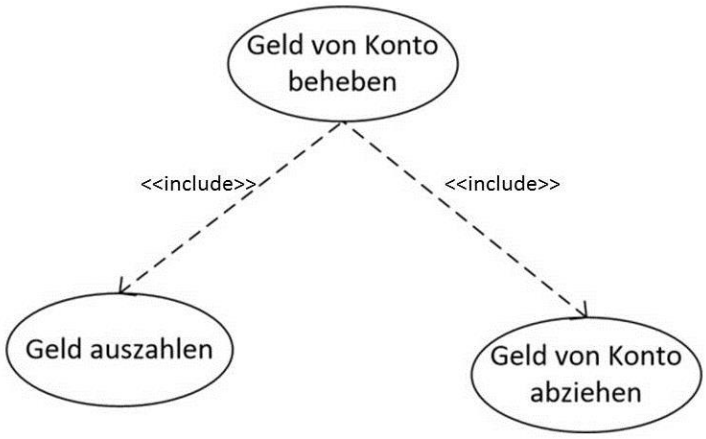


Abbildung 29 zeigt den Anwendungsfall „Geld von Konto beheben“ der 2 weitere Anwendungsfälle inkludiert („Geld auszahlen“ & „Geld von Konto beheben“)

- Extend: die Extend-Beziehung erweitert einen Anwendungsfall. Wie in der Abbildung 30 gezeigt, kann man „Geld von Konto beheben“ nicht durchführen, ohne „Konto anmelden“ durchgeführt zu haben.

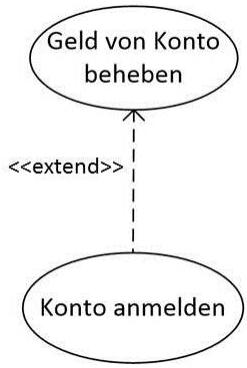


Abbildung 30 zeigt 2 Anwendungsfälle, wobei der Anwendungsfall „Geld von Konto beheben“ den Anwendungsfall „Konto anmelden“ voraussetzt.

3.2.3.4. Adaption

Das Use Case Modell ist bereits ein relativ übersichtliches Modell und kann, falls es nicht zu viele Elemente umfasst, vermutlich problemlos auf jedem Smartphone angezeigt werden.

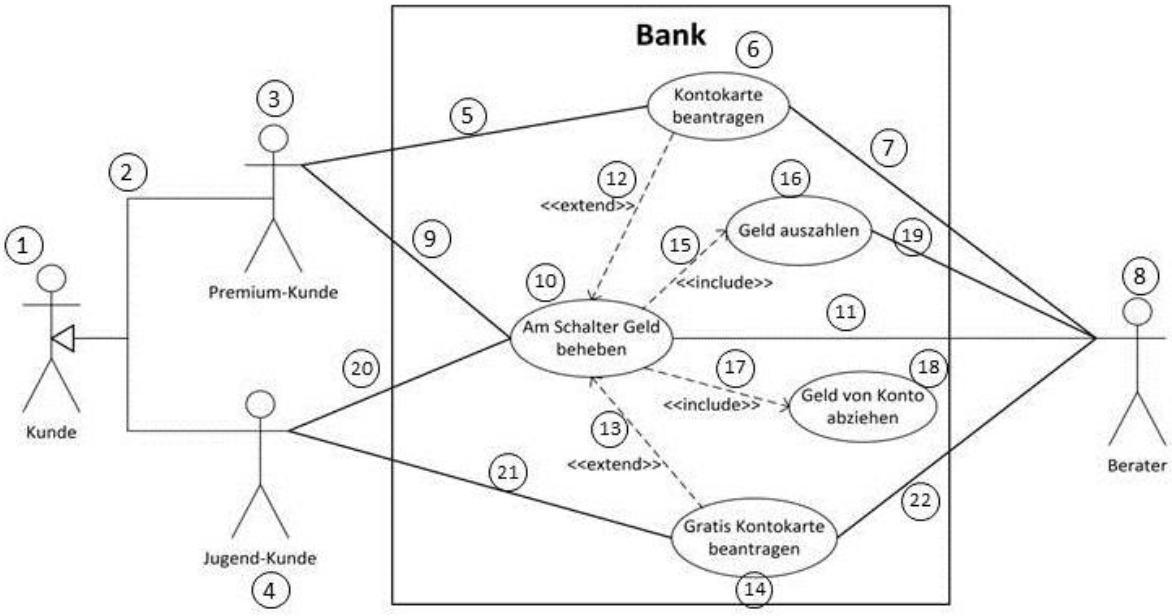


Abbildung 31 zeigt das Beispiel eines Anwendungsfalldiagramms, das verschiedene Anwendungen einer Bank darstellt. Auf der linken Seite stehen die Kunden, die über die Anwendungsfälle in der Mitte mit dem Berater auf der rechten Seite interagieren können.

Wie bereits am Anfang des Kapitels beschrieben, kann es für Benutzer sehr interessant sein, nur einen Teil des Modells zu sehen. Im Falle des Anwendungsfalldiagramms kann dies auch sehr einfach umgesetzt werden. Im Folgenden werden nun kurz die Vor- und Nachteile näher beschrieben und ein Resümee gezogen:

- **Textuell**

Eine textuelle Beschreibung eines Use Case Diagramms wäre durchaus möglich. Durch die Hilfe von Textbausteinen könnte man jeden Akteur als „Benutzer“ und jede Anwendung als „Anwendung“ beschreiben. Die Beziehungen hingegen würden ihren Namen behalten. Dadurch könnte das Beispiel aus Abbildung 31 in einige Textbausteine aufgeteilt werden. Man kann in Abbildung 31 die gelben Punkte mit den Nummern aus Abbildung 32 den Textbausteinen zuordnen.

Abbildung 32 zeigt Textbausteine mit Verweisen in den gelben Kreisen zu Abbildung 31 für dasselbe Anwendungsfalldiagramm

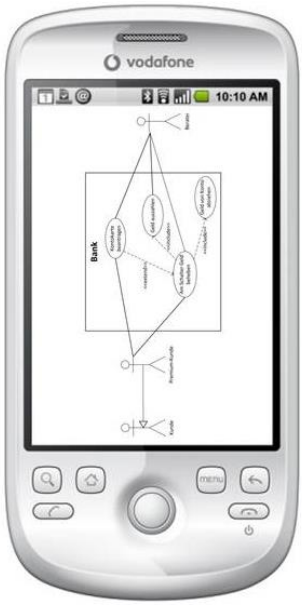
Wie man hier sehr schnell erkennen kann, wird ein kleines Modell mit nur 9 Elementen und 13 Beziehungen in der textuellen Darstellung unübersichtlich. Durch das

Hinzufügen von Farben kann man zumindest die Beziehungen zwischen den Elementen besser darstellen und übersichtlicher für den Benutzer gestalten. Weiters könnte man das Modell einschränken und die Sicht auf einen Benutzer oder einen Anwendungsfall zentrieren. Das heißt, es würden nur alle Beziehungen und Elemente, die mit dem zentrierten Element verbunden sind, aufgezählt werden. Damit könnte man den Text etwas verkürzen und so etwas mehr Übersichtlichkeit schaffen. Um ein Element auszuwählen, könnte man den Benutzer am Smartphone zuerst fragen, ob die Sicht auf einen Anwendungsfall oder auf einen Benutzer zentriert werden soll. Nach dieser Entscheidung würden dann alle Elemente der Auswahl aufgezählt werden. Aus dieser Liste kann dann das zentrale Element ausgewählt werden und die Darstellung in textueller Form rund um das ausgewählte Element aufgebaut werden.

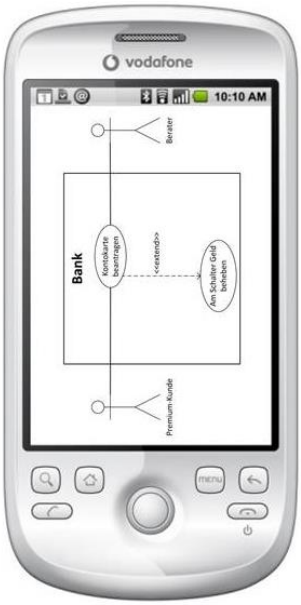
- **Graphisch**

Das Modell kann, falls es nicht zu groß ist, normal am Smartphone Display angezeigt werden. Wie bereits auch in der textuellen Sicht erwähnt, könnte der Benutzer eine zentrierte Sicht wählen, auf der der Fokus auf einen speziellen Akteur gelegt wird und nur die Anwendungen und Beziehungen dieses Akteurs angezeigt werden. Der Benutzer könnte aber auch hier statt einem Benutzer einen Anwendungsfall auswählen und der Fokus würde dann auf diesem Anwendungsfall liegen. Der User des Smartphones könnte im Beispiel des Akteurs eine Liste mit allen Akteuren erhalten und sich mithilfe eines Radiobutton[17] für einen dieser Akteure entscheiden. Im Fall der Abbildung 31 würde dann einfach ein Teil des Modells wegfallen, dies ist in Abbildung 33 veranschaulicht:

[17] Ein Radiobutton ist ein sogenanntes „Optionsfeld“ mit dem man genau eine Option aus mehreren Möglichkeiten auswählen kann. (Android 2)



Zentrierte-Sicht:
„Premium-Kunde“



Zentrierte-Sicht:
„Kontokarte beantragen“

Abbildung 33 zeigt die zentrierte Sicht auf den Akteur „Premium-Kunde“ (linke Abbildung) und die zentrierte Sicht auf den Anwendungsfall „Kontokarte beantragen“ (rechte Abbildung) (Abbildung des Smartphones Zitat (HTC Magic))

Man kann auf den zwei Abbildungen gut erkennen, was abgebildet wird, und könnte bei Bedarf durch den Multitouch-Bildschirm das ausgeschnittene Modell vergrößern. Die zentrierte Sicht auf den Akteur „Premium-Kunde“ enthält beinahe alle Elemente des Modells. Es wurden insgesamt nur 2 Elemente und 3 Beziehungen entfernt. Da das Anwendungsfallmodell relativ klein ist, könnte man dem Benutzer des Smartphones auch das komplette Modell zeigen. Es würde nicht zu unübersichtlich wirken und durch den Touchscreen und die Zoomfunktion wäre das ganze Modell auch gut lesbar.

- **Vergleich**

Die Anpassungen der Modellnotationen können wie folgt aussehen:

- Graphische Aufbereitung: Wird mit der zentrierten Sicht und der Reduktion abgedeckt.
- Zentrierte Ansicht: Bei der zentrierten Ansicht wählt der Benutzer einen Akteur aus, rund um diesen werden alle Beziehungen und Anwendungsfälle geladen und angezeigt.

- Textuell: Das Modell wird textuell beschrieben. Eine Kombination mit der zentrierten Ansicht macht auch hier wieder durchaus Sinn.
- Farblich: Beziehungen und zusammenhängende Anwendungsfälle können sowohl graphisch als auch textuell farblich dargestellt werden.
- Reduktion von Informationen: Man kann die Reduktion von Informationen mit der zentrierten Ansicht kombinieren und zusätzlich wieder einen „+“ und einen „-“ Knopf anzeigen. Damit können Vererbungen angezeigt oder ausgeblendet werden. Jeder User der von einem anderen User abgeleitet wird, bekommt ein „+“ Symbol, um anzuzeigen, dass dieser User eine Generalisierungsbeziehung versteckt hat.

Der direkte Vergleich zwischen den unterschiedlichen Darstellungsarten:

Abbildung 34 zeigt am linken Smartphone ein Modell in der unveränderten Notation, das Smartphone in der Mitte zeigt die textuelle Beschreibung und das Smartphone auf der rechten Seite zeigt ein Modell mit einer zentrierten Ansicht auf den User „Premium Kunde“ (Abbildung des Smartphones Zitat (HTC Magic))

Die farbliche Kennzeichnung der Beziehungen in der textuellen Ansicht macht den Text ein wenig übersichtlicher. Die zentrierte Ansicht des Modells zeigt einen sehr übersichtlichen Ausschnitt aus dem Modell.

Die textuelle Form kann in diesem Fall leider nur durch Unübersichtlichkeit (siehe Abbildung 32) glänzen und verwirrt den Benutzer durch die viele Beziehungen, die zu einem Anwendungsfall führen oder von diesem wegführen. 13 Beziehungen scheinen zu viel für die textuelle Beschreibung zu sein und eines der Hauptprobleme mit den Beziehungen ist, dass es keine „Fließrichtung“ im Diagramm gibt. Man kann nicht einfach ein Element nach dem anderen aufzählen und die Beziehung dazwischen benennen, wie es zum Beispiel bei einem Aktivitätsdiagramm oder bei einem Zustandsdiagramm möglich wäre. Viele Anwendungsfälle haben mehrere Beziehungen zu anderen Akteuren oder Anwendungsfällen, dadurch wird es sehr schwierig zu lesen und man kann sich das Modell nur schwer vorstellen.

Die graphische Darstellung behält die Übersichtlichkeit bei und auch bei 10 Beziehungen, wie in Abbildung 33, ist das Modell noch gut zu erkennen. Man kann die Beziehungen noch leicht verfolgen und erkennen, was das Modell darstellen soll.

Durch die Übersichtlichkeit des Modells und der Möglichkeit einer Akteur- oder Anwendungsfall-Spezifischen-Sicht ist hier die graphische Darstellung der textuellen eindeutig überlegen.

4. Kommunikationsprotokolle und Kommunikationstechnologien

Um ein Modell mobil auf einen Smartphone anzeigen zu können, müssen eine Vielzahl Daten aus dem Internet geladen werden. Wie im letzten Kapitel bereits erklärt wurde, werden einige Daten in den Modellen erst nach ausdrücklichem Benutzerwunsch angezeigt. Damit man beim initialen Laden der Daten nicht so lange warten muss, werden nur die wichtigsten Daten geladen.

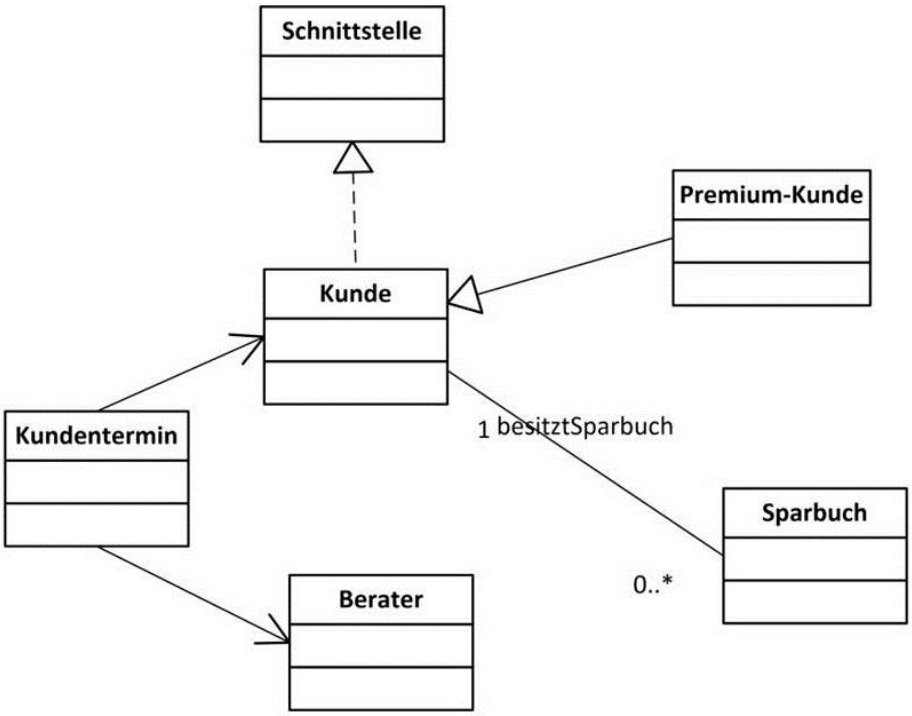


Abbildung 35 zeigt ein Klassendiagramm mit reduzierten Informationen

Im Fall der Abbildung 35 müssen 6 Klassen und die dazugehörigen Beziehungen geladen werden. Nachdem man eine Übersicht über das Klassendiagramm bekommen hat kann man für jede Klasse Attribute und Methoden nachladen. Die reduzierten Informationen, die erst nach Benutzerwunsch angezeigt werden, werden erst auf Nachfrage geladen (das Prinzip von „Lazy Loading“). Der Benutzer erwartet sich einen flüssigen Arbeitsbetrieb und dass Modelle in angemessener Zeit auf dem Display seines Devices auftauchen. Diese Erwartung ist unabhängig von seinem Standort und der dort verfügbaren Netzleistung des Mobilfunkanbieters.

Ein Kommunikationsprotokoll ist eine „Verhaltenskonvention, die die zeitliche Abfolge der Interaktionen zwischen den diensterbringenden Instanzen vorschreibt und die Formate (Syntax und Semantik) der auszutauschenden Nachrichten definiert“. (Zitat (Protocol Engineerin) Seite 26 Absatz 2) Das heißt, ein Kommunikationsprotokoll beinhaltet alles, was benötigt wird, damit zwei (oder mehr) Parteien miteinander kommunizieren können. Kommunikationsprotokolle stellen den Grundstein für Kommunikationstechnologien dar.

Kommunikationstechnologien sind Technologien, mit denen man meist elektronische Nachrichten austauschen kann. Zumeist kommen diese bei der Telekommunikation, Mobilkommunikation oder auch Satellitenkommunikation zum Einsatz. Die Kommunikationstechnologie beinhaltet auch die Netzwerktechnologien und die Verbindungstechnologien, die zum Einsatz kommen, um die Kommunikation über weite Strecken zu ermöglichen. Diese Distanzen werden in der heutigen Zeit zumeist über Funkfrequenzen und nicht über Kabel überwunden. Mit Hilfe von Kommunikationstechnologien werden Daten, die zur Anzeige eines Modells benötigt werden, übertragen.

In den folgenden Unterkapiteln werden unterschiedliche Technologien und Möglichkeiten beschrieben, wie man eine Applikation optimal mit Daten befüllen kann, um Modelle auf kleinen Devices gut erkennbar anzuzeigen. Es soll sichergestellt werden, dass ein Modell möglichst schnell auf einem Device angezeigt werden kann. Dies gilt auch für Informationen, die der Benutzer auf Wunsch nachladen möchte.

4.1. Kommunikationsdienst

Ein Provider stellt einen Dienst zur Verfügung, der zur Kommunikation genutzt werden kann. Wie so ein Kommunikationsdienst aussehen kann, wird in folgender Abbildung dargestellt:

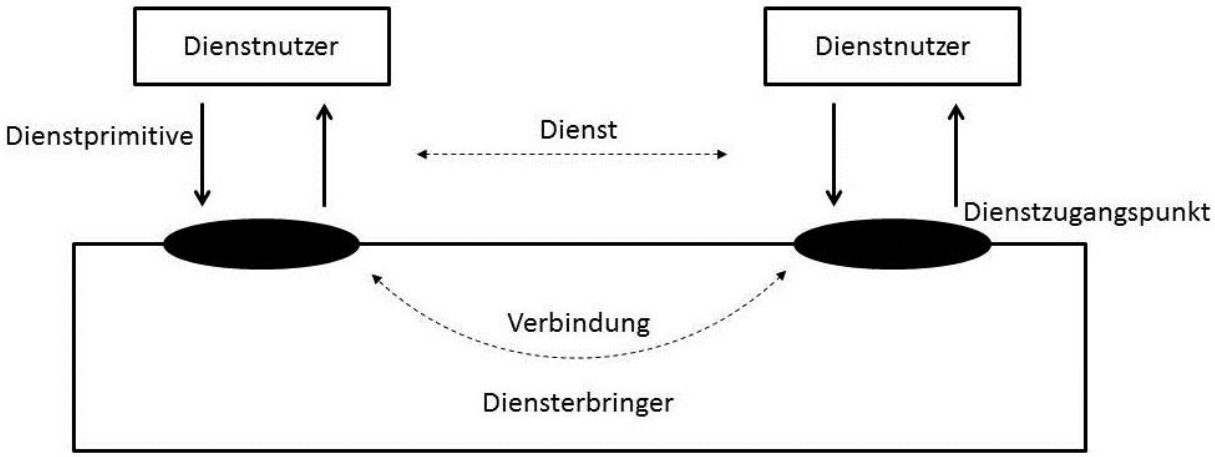


Abbildung 36 zeigt einen Kommunikationsdienst zwischen Dienstnutzern und einem Diensterbringer. (Zitat von (Protocol Engineering) Abbildung 1. 1/3 Seite 5)

Bei einem Kommunikationsdienst wird zwischen zwei Rollen unterschieden:

- Dienstnutzer (englisch: service user): wie der Name schon sagt, kann der Dienstnutzer den Dienst des Dienstbringers benutzen.
- Dienstbringer (englisch: service provider): stellt einen Dienst für alle Dienstnutzer zur Verfügung

Um einen Dienst in Anspruch nehmen zu können, sind immer mindestens zwei Dienstnutzer notwendig. Das Beispiel aus der Quelle (Protocol Engineering) ist das Telefonnetz. Der Dienstanbieter bietet allen Dienstnutzern das Telefonnetz zur Benutzung an. Ein Dienstnutzer kann dies aber nur dann verwenden, wenn ein anderer Nutzer auf seinen Anruf reagiert. Die Telefonkommunikation kann nur zwischen zwei Telefonnutzern stattfinden.

Ein Dienstnutzer kann über einen sogenannten Dienstzugangspunkt (englisch: service access point) auf einen Dienst zugreifen. Im oben genannten Beispiel kann dies ein Telefonapparat oder auch ein Mobiltelefon sein.

Ein Dienstprimitive (englisch: service primitive) beschreibt, wie ein Benutzer einen Dienst verwenden kann. Ein Dienstprimitive wird durch drei Punkte charakterisiert:

- Name
- Typ
- Parameter

Der Name spiegelt die allgemeine Funktion des Dienstes wieder. In der Quelle (Protocol Engineering) werden folgende Beispiele angeführt: „

- CONNECT - Verbindungsaufbau
- DISCONNECT - Verbindungsabbau
- DATA - Datentransfer
- ABORT - Verbindungsabbruch

“ (Zitat (Protocol Engineering) Seite 6 Absatz 5)

Der Typ gibt an, welche Funktion verwendet wird, und man unterscheidet im Allgemeinen zwischen vier unterschiedlichen Typen:

- Request (deutsch: Anfrage): ist eine Anfrage vom Benutzer an den Dienstbringer durch den Dienstzugangspunkt.
- Respond (deutsch: Antwort): ist eine Antwort an den Benutzer auf dem Dienstzugangspunkt.
- Indikation (deutsch: Anzeige): wird am zugehörigen „Partner-Dienstzugangspunkt“ angezeigt. Man kann sich hier am einfachsten die Kommunikation über zwei Mobiltelefone vorstellen. Der Dienstbringer zeigt dem angerufenen Benutzer die Nummer des Anrufers auf dessen Mobiltelefon an.
- Confirm (deutsch: Bestätigung): ist die Bestätigung wird dem Benutzer am Anfragenden-Dienstzugangspunkt gegeben. Im vorherigen Beispiel kann dies das Anrufzeichen beim Anrufer sein.

Der Parameter enthält alle wichtigen Daten, die zur Ausführung des Dienstes notwendig sind.

Die Abfolge der Aufrufe der Dienstprimitiven ist nicht zufällig, sondern es gibt immer eine Reihenfolge, die eingehalten werden muss. Wenn man beim Beispiel des Anrufs bleibt, muss man auch zuerst den Hörer abnehmen, auf das Freizeichen warten (Dienstzugangspunkt), Nummer wählen (Dienstprimititive Request), es läutet (Dienstprimititive Confirm) und dann wird der angerufene Benutzer entweder abheben oder nicht.

Dieses Kapitel war als Übersicht gedacht, damit der Leser einen Überblick bekommt, wie man sich einen Dienst vorstellen kann und wie die interne Kommunikation in solchen

Fällen aussieht. Im Folgenden werden nun Protokolle an sich erklärt, wie diese mit dem Kommunikationsdienst interagieren und wie diese verwendet werden.

4.2. Protokolle

Ein Protokoll hat in der deutschen Sprache zwei Bedeutungen: das Aufzeichnen einer Besprechung oder den festen Ablauf einer Tätigkeit. Im Falle dieser Arbeit wird das Protokoll als Ablauf einer Tätigkeit angesehen. Das Protokoll in der Informatik umfasst Regeln, Syntax, Synchronisation und Semantik. Diese vier Punkte sind unbedingt notwendig, damit eine Kommunikation zustande kommen und interpretiert werden kann. Das Kommunikationsprotokoll kann auf Hardware, Software oder einer Verbindung aus beiden implementiert werden.

In der Abbildung 36 wird beschrieben, dass der Diensterbringer eine Verbindung zwischen den beiden Dienstzugangspunkten zur Verfügung stellt. Diese Verbindung wird in folgender Abbildung noch einmal genauer dargestellt:

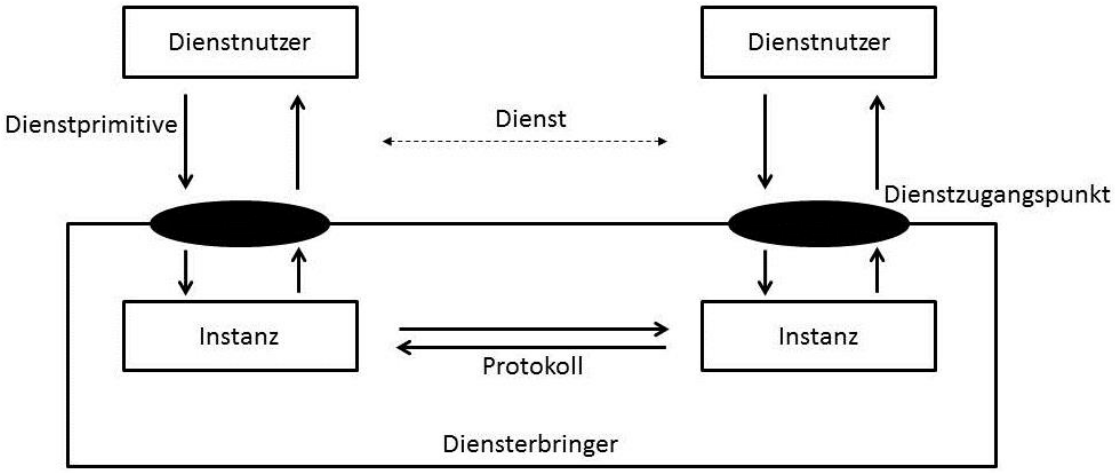


Abbildung 37 zeigt wie der Diensterbringer die Verbindung zwischen den beiden Dienstzugangspunkten ermöglicht (Zitat von (Protocol Engineering) Abbildung 2. 1/1 Seite 25)

Die Diensterbringer ermöglicht die Verbindung der Dienstnutzer mit Hilfe von Instanzen. Diese sind aktive Objekte des Diensterbringers, die einen Austausch von Nachrichten ermöglichen. Die Interaktion zwischen den Instanzen übernimmt ein Protokoll. Eine Instanz ist in hierarchische Schichten unterteilt. Jede Schicht stellt einen

oder mehrere Dienste zur Verfügung und jede Schicht verwendet nur Dienste von der nächst unterlegenen Schicht, um zu kommunizieren. Das folgende Beispiel beschreibt die Architektur der Schichten am besten:

Ein Philosoph aus Afrika möchte sich mit einem Philosophen aus Asien über eine philosophische These unterhalten. Der afrikanische Philosoph schreibt seinen Text in seiner Muttersprache Suaheli. Diesen Text gibt er seiner untergeordneten Schicht, um diesen übersetzen zu lassen. Der Übersetzer muss mit dem Übersetzer aus Asien korrespondieren, um sich eine Sprache auszumachen, in die der Text übersetzt werden soll. Die Korrespondenz wird durch die untergeordnete Schicht des Technikers übernommen. Der Techniker stellt die Verbindung zu einem Techniker in Asien her, um die Frage der Übersetzungssprache dem anderen Übersetzer zu übermitteln. Dieser antwortet, dass er die englische Sprache bevorzugen würde und lässt die Antwort mittels der untergeordneten Schicht des asiatischen Technikers übermitteln. Der afrikanische Techniker übermittelt die Antwort an die übergeordnete Schicht und der Übersetzer kann den Text von Suaheli auf Englisch übersetzen. Danach wird der Text an den Techniker übergeben, der diesen an den asiatischen Techniker übermittelt. Dieser händigt den Text an den asiatischen Übersetzer aus, der den Text aus dem Englischen ins Indonesische übersetzt. Danach wird der übersetzte Text an den asiatischen Philosophen vom Übersetzer weitergereicht.

Dieses Beispiel kommt aus der Quelle (Computernetzwerke) und erklärt die Schichtenarchitektur am anschaulichsten. Diese Schichtenarchitekturen werden vom Kommunikationssystem aufeinander abgestimmt und Kommunikationsprotokolle werden darin eingebettet. Dies wird auch als Kommunikationsarchitektur bezeichnet und umfasst die Funktionen der Schichten und wie diese miteinander interagieren.

Ein Dienstzugangspunkt kann immer nur einer Instanz zugeordnet sein. Eine Instanz kann jedoch mehrere Dienstzugangspunkte abfertigen. In der Quelle (Protocol Engineering) wird eine Instanz mit einem Briefverteilerzentrum verglichen. Die Instanzen leeren alle Briefkästen, die ihnen zugeordnet sind, aus und kümmern sich darum, dass die Briefe ins Verteilerzentrum kommen. Das Verteilerzentrum übernimmt dann den Teil der Arbeit, so dass die Briefe in die richtigen Postkästen zugestellt werden. Die Interaktion zwischen den Instanzen wird von Regeln eines Protokolls bestimmt. Wie die Instanzen mit einem Protokoll kommunizieren, wird in der

Protokollspezifikation festgelegt. In dieser Spezifikation wird beschrieben, wie die Instanzen zeitlich miteinander interagieren. Des Weiteren ist festgelegt, wie Instanzen auf Daten reagieren und vor allem in welchem Format die Daten ausgetauscht werden. Nachrichten, die von den Instanzen ausgetauscht werden, heißen Protokolldateneinheiten (PDUs). „Die PDUs bilden quasi die „gemeinsame Sprache“ der kommunizierenden Instanzen, die von beiden Seiten gleich interpretiert werden. Folglich müssen in der Spezifikation die Formate eindeutig festgelegt werden.“ (Zitat: (Protocol Engineering) Seite 31 Absatz 1) Die PDUs werden auf Seiten des Senders kodiert und auf Seiten des Empfängers analysiert und dekodiert. Diese Aufgaben machen einen großen Teil des Protokoll-Codes aus. Die Struktur wird durch das Protokoll festgelegt und ist der jeweiligen Kommunikationspartner-Instanz bekannt. Damit können alle PDUs gleich interpretiert werden. Man kann meistens von drei unterschiedlichen Strukturen ausgehen wie etwa: eine Struktur für den Verbindungsaufbau, eine Struktur für den Datentransfer und eine Struktur für den Verbindungsabbau. Der Dienstnutzer selbst benötigt keinerlei Informationen, wie die Instanzen miteinander kommunizieren oder auf welchen Regeln ein Protokoll basiert. Dies wird alles vom Diensterbringer realisiert und die Protokollspezifikation ist implementierungsunabhängig definiert.

Man unterscheidet zwischen zwei unterschiedlichen Protokollen:

Symmetrische Protokolle:

Ein symmetrisches Protokoll ist dann gegeben, wenn beide Instanzen dasselbe Kommunikationsverhalten an den Tag legen und das Protokoll einen duplexen Datenaustausch zulässt. Dabei werden Daten gleichzeitig in beide Richtungen geschickt und empfangen.

Asymmetrische Protokolle:

Bei einem asymmetrischen Protokoll unterscheiden sich die beiden Instanzen im Verhalten. Hier kann es zu einer simplexen Datenübertragung kommen, das heißt es werden Daten nur in eine Richtung übertragen.

Der Diensterbringer kann die Daten, die mittels eines Protokolls von einer zur anderen Instanz transportiert werden, nicht einsehen. Hier gilt ein elektronisches „Briefgeheimnis“. Dieses wird auch als Prinzip der Transparenz bezeichnet. Bei diesem

Prinzip weiß man zwar, dass ein Teil eines Systems vorhanden ist, jedoch ist es nicht einsehbar.

Jedes Protokoll hat gewisse Funktionen, die von den Instanzen genutzt werden, um die Protokollabläufe abzuwickeln. Einige dieser Funktionen werden von fast allen Protokollen, wenn auch in veränderter Version, zur Verfügung gestellt. Diese werden in den folgenden Unterkapiteln erklärt.

4.2.1. Verbindungsverwaltung

Die Verbindungsverwaltung ist zuständig für den Aufbau, Verwaltung und den Abbau einer Verbindung zwischen Schichten. Der Verbindungsaufbau muss die Partner-Instanzen synchronisieren, um zu kontrollieren, ob die Verbindung angenommen oder abgelehnt wird. Dies passiert mit Handshake-Verfahren. Beim Handshake wird ein Verbindungsaufbauwunsch durch die Partner-Instanz bestätigt.

Der Verbindungsaufbau kann auf zwei Methoden passieren:

- Explizt: Die Datenübertragung wird erst nach einer positiven Bestätigung des Verbindungsaufbaus durchgeführt.
- Implizit: Die Datenübertragung wird vor der positiven Bestätigung bereits gestartet. Diese Art des Verbindungsaufbaus wird aber meistens bei Hochleistungsprotokollen benutzt.

Die Datenübertragung ist für den Dienstnutzer meist unbekannt und kümmert sich, was passieren soll, wenn eine Verbindung zusammenbricht. Beim Verlust der Verbindung gibt es zwei Möglichkeiten, wie man mit der Datenübertragung fortfahren soll:

- Wiederaufsetzen: Die Verbindung wird erneuert.
- Neu zuweisen: Die Verbindung wird neu zugewiesen.

Der Verbindungsabbau wird meist nach der Datenübertragung durchgeführt. Es kann aber auch passieren, dass die Verbindung wegen technischer Probleme beendet wird. Beim Verbindungsabbau kann es, so wie beim Verbindungsaufbau und auch bei der Datenübertragung, zwei Methoden geben:

- Explizit: Diese Methode wird nach der erfolgreichen Datenübertragung durchgeführt und die Verbindung wird abgebaut.
- Abrupt: Die Verbindung wird sofort abgebaut (abgebrochen), egal wie weit die Datenübertragung fortgeschritten ist. Bei einem Abbruch kann es passieren, dass die Verbindungsreferenz erneut vergeben wurde, bevor die Datenübertragung bemerkt hat, dass sich der Partner geändert hat. In diesem Fall würden dem neuen Partner Daten zugeführt werden, die nicht für ihn bestimmt sind. Um dieses Problem zu umgehen, werden abgebrochenen Verbindungsreferenzen für einen bestimmten Zeitraum nicht wieder vergeben. (Protocol Engineering) (Computernetzwerke)

4.2.2. Synchronisation

Die Synchronisation ist sehr wichtig für die Verbindungsverwaltung. Vor dem Verbindungsaufbau sollten beide Seiten über den Aufbau informiert werden und beim Abbau ist es, auch aus Datensicherheitsgründen, überaus wichtig, dass beide Seiten die Verbindung wieder abbauen. Daher müssen Instanzen synchronisiert werden. Zu Problemen kann es bei der Synchronisation kommen, wenn Protokolldateneinheiten, die für die Synchronisation notwendig sind, zu spät kommen oder verloren werden. (Protocol Engineering)

4.2.3. PDU-Kodierung/-Dekodierung

Die PDU Kodierung passiert beim Sender und die Dekodierung beim Empfänger. Diese Operationen können einen wesentlichen Teil der Protokollaktivitäten ausmachen und daher die Datenübertragung verlangsamen, wenn nicht effektiv gearbeitet wird. Die Instanz übernimmt die Kodierung bzw. Dekodierung und das Protokoll muss die PDU mit Informationen versorgen. Das Protokoll muss sich auch um die Weiterleitung der PDUs kümmern. Dies passiert meist in Puffern, die von Schicht zu Schicht weitergegeben werden. (Protocol Engineering)

4.2.4. Fehlerkontrolle

Die Fehlerkontrolle soll alle Fehler erkennen und beheben, die bei einer Interaktion zwischen Instanzen auftauchen können. Folgende Fehler können immer wieder auftauchen:

- Cyclic Redundancy Checks: Der Sender erzeugt eine Prüfsumme, die einer PDU hinzugefügt wird und ohne Rest durch einen bestimmten Wert teilbar ist. Der Empfänger dividiert die angehängte Information mit demselben Wert und wenn der Rest 0 beträgt, ist die Übertragung ohne Fehler bzw. ohne Manipulation übertragen worden.
- Bestätigung: Der Empfänger sendet nach der erfolgreichen Übertragung eine Bestätigung an den Sender. Diese Bestätigung kann positiv oder negativ ausfallen.
- Zeitüberwachung: Ein klassisches Fehlerszenario ist, dass die Zeit gestoppt wird, nachdem der Start der Übertragung begonnen hat. Falls ein Fehler auftritt, wird keine Erfolgsmeldung zurückgeschickt und die Zeit wird ablaufen. Dadurch wird ein Timeout ausgelöst. Dies soll zum Beispiel einen Deadlock[18] verhindern oder einen Verbindungsverlust abfangen.
- PDU-Verluste: Zu einem PDU-Verlust kann es kommen, falls eine PDU nicht rechtzeitig oder bereits eine PDU mit einer höheren Sequenz- oder Positionsnummer eintrifft. Weiters kann das Ausbleiben einer Bestätigung bereits ein Indiz für einen PDU-Verlust sein.
- PDU–Dopplungen: Eine PDU-Dopplung kann von der Instanz mit Hilfe der Sequenznummer oder Positionsnummer erkannt und ignoriert werden.

4.3. Webservice

Um Daten für Modelle immer und überall mobil abrufen zu können, muss man diese über das WWW versenden und empfangen können. Für diese Anforderung eignen sich am besten Webservices. Ein Webservice dient laut Quelle (W3C Web Service) "to support interoperable machine-to-machine interaction over a network" (deutsche Übersetzung: "um interoperable Maschine-zu-Maschine Interaktionen über ein Netzwerk zu unterstützen) (Zitat (W3C Web Service) Kapitel 2 Absatz "Web service").

[18] Ein Deadlock in der Informatik bedeutet, dass ein Prozess auf einen anderen wartet und vice versa. Beide Prozesse warten darauf, dass der andere seinen Zustand wechselt.

Ein Webservice soll also die Kommunikation zwischen zwei Maschinen unterstützen, daher muss auch die Kommunikation nur für Maschinen und nicht für den Benutzer lesbar sein. Das Ganze geschieht über ein Netzwerk, wie zum Beispiel das Internet. (Java Web Services) In der Quelle (W3C Web Service) aus dem Jahr 2004 wird nur eine Methode zum Ansprechen des Webservices beschrieben. Dies geschieht mit Hilfe einer SOAP-Nachricht. Man kann aber auch durch einen „Remote Procedure Call“ (RPC) [19] ein Webservice aufrufen. (Einstieg in XML) Weiters gibt es die Möglichkeit ein Webservice, zum Beispiel "RESTful", zur Verfügung zu stellen, das man mit der REST-Technologie ansprechen kann. Im folgenden Kapitel 4.4 werden diese Methoden verglichen und eine Empfehlung für den effektiven Aufruf eines Webservices mit einer Smartphone Applikation abgegeben. (REST) (Webservices mit REST)

Nach der Quelle (W3C Web Service) der W3C hat ein so definiertes Webservice ein Interface implementiert, das in der Sprache Web Services Description Language (WSDL) geschrieben ist. WSDL ist von der W3C als Recommendation in der Version 2.0 im Jahr 2007 veröffentlich worden. [20] WSDL ist eine Beschreibungssprache, die auf XML basiert und dazu dient, Nachrichten über ein Netzwerk bzw. das Internet plattformunabhängig auszutauschen. Mit Hilfe eines WSDL-Dokuments wird ein Webservice beschrieben. In so einem Dokument wird die Funktionalität des Webservices definiert und alle benötigten Informationen, wie Input- und Output-Daten, hinterlegt. Weiters liefert die WSDL-Datei wichtige Informationen über die Schnittstelle. Die WSDL-Datei verrät dem Anwender, ob Input-Parameter notwendig oder optional sind, sowie in welchem Format und in welcher Reihenfolge das Webservice diese Parameter erwartet. Ein WSDL-Dokument besteht aus sechs unterschiedlichen Elementen: (W3C WSDL 2.0) (Einstieg in XML)

- Types: Enthält alle Datentyp-Definitionen, die für den plattformunabhängigen Nachrichtenaustausch notwendig sind. Zumeist wird hier in den types-Tags das XML Schema (XSD)[21] zum Einsatz kommen.

[19] (deutsch: Ferner Aufruf einer Prozedur) ermöglicht entfernte Prozeduren auf anderen Computern oder Servern aufzurufen bzw. auszuführen.
[20] Es gibt auch einen Standard, der im Jahr 2001 unter der Version 1.1 (W3C WSDL 1.1) veröffentlicht wurde
[21] XML Schema Definition ist eine Recommendation der W3C aus dem Jahr 2001 und dient zur Strukturierung von XML Dokumenten. Das Schema enthält Regeln wie ein XML Dokument aufgebaut sein darf. (Einstieg in XML) Nähere Informationen dazu siehe Kapitel 4.5.2 dieser Arbeit.

- Message: Eine Message (Nachricht) besteht aus einem oder mehreren logischen Parts (Teilen), die Variablennamen und Datentyp-Definitionen enthalten.
- PortType: enthält die Namen von abstrakten Operationen, die zur Verfügung stehen. Es gibt vier unterschiedliche Arten von Operationen: One-Way (Einbahn: Der Endpunkt erhält eine Nachricht), Request-Respons (Nachfrage-Antwort: Der Endpunkt erhält eine Nachricht und schickt eine Nachricht zurück), Solicit-response (Erbetene Antwort: Der Endpunkt versendet eine Nachricht und bekommt eine Nachricht zurück), Notification (Benachrichtigung: Der Endpunkt versendet eine Nachricht)
- Binding: Das Binding (Bindung) beschreibt das Nachrichtenformat und das Nachrichtenprotokoll.
- Port: Der Port definiert eine eindeutige Adresse für ein Binding.
- Service: Ein Service gruppiert eine beliebige Anzahl an zusammengehörigen Ports.

(W3C WSDL 1.1) Ein Beispiel für eine WSDL-Datei wird in Kapitel 4.4.1 beschrieben, wo auch auf eine WSDL-Datei eingegangen wird, die von einem Webservice zur Verfügung gestellt wird.

Der Transport der SOAP-Nachricht wird über HTTP erfüllt und enthält ein XML mit Informationen für die anfragende Maschine. In Abbildung 38 kann man das klassische Webservice sehen, wie es anfangs geplant war:

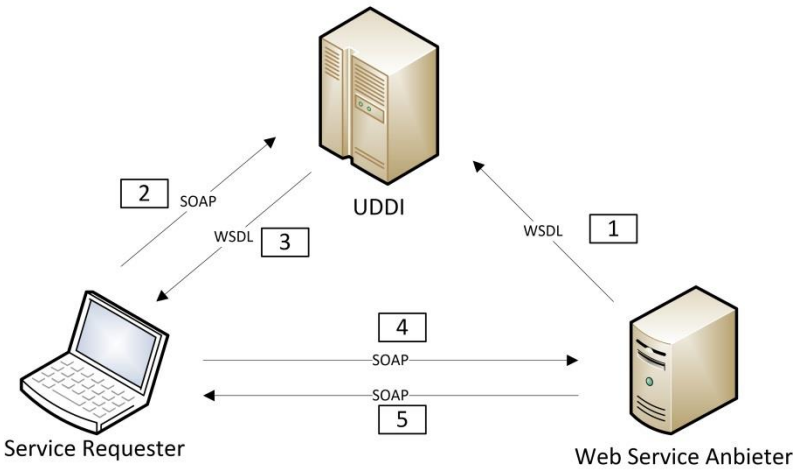


Abbildung 38 zeigt ein typisches Webservice Scenario mithilfe der Protokollsprache SOAP. (XML WS)

Der erste Schritt (1) ist die Registrierung des Services bei der Universal Description, Discovery and Integration (UDDI). Der nächste Schritt ist die Anfrage (2) des Service Requesters bei der UDDI und die Antwort (3) mit den nötigen Informationen, um das Service aufrufen zu können. Danach ruft der Service Requester das Webservice mittels SOAP-Nachricht und den benötigten Attributen auf (4) und erhält von diesem Service eine SOAP-Nachricht mit den angefragten Informationen zurück (5). (Einstieg in XML)

Eine UDDI dient dazu, dass Entwickler ihre Webservices registrieren und für Anwender zur Verfügung stellen können. Der Anwender kann in dem Verzeichnis UDDI nach unterschiedlichen Webservices suchen und diese dann verwenden. In der heutigen Zeit ist jedoch die UDDI-Technologie überholt und wird nicht mehr verwendet (UDDI InfoWorld) (UDDI ZDNet). Man kann also die Verbindungen zum Knoten UDDI streichen und es bleibt dann eine einfache Grafik mit einer Anfrage und einer Antwort übrig:

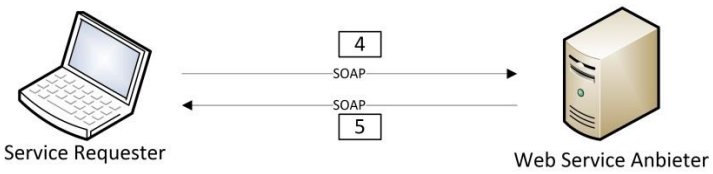


Abbildung 39 zeigt ein typisches Webservice in der heutigen Zeit ohne die Komponente UDDI.

Der Nachteil, der sich dadurch ergibt, ist, dass der Aufrufer des Webservices wissen muss, wie die WSDL-Datei des Services aussieht, um diese mit den notwendigen Variablen befüllen zu können. Ein anderer großer Vorteil, der mit der Beendigung der UDDI verloren ging, ist die Möglichkeit nach Webservices zu suchen. Man konnte beim UDDI ein Webservice registrieren und dann konnten Benutzer dieses Service finden und die WSDL-Datei einsehen. Damit konnte man auch unbekannte Webservices nutzen und ansprechen. Ohne diesen Service muss nun jeder Nutzer eines Webservices vorher wissen, dass es das Webservice gibt und wo man es ansprechen kann (d.h. die einzigartige Uniform Resource Identifier (URI)[22] muss dem Nutzer bekannt sein).

[22] Eine „Uniform Resource Identifier“ (deutsch: einheitlicher Ressource Identifikator) ist eine einheitliche Form Ressourcen zu identifizieren. Mit Hilfe der URI werden Ressourcen einheitlich und vor allem einzigartig umschrieben, um diese adressieren zu können. Eine URI wird zum Beispiel im Internet verwendet, um Webseiten zu adressieren. In diesem Fall ist die URI entweder eine Uniform Resource

Ein Webservice kann also von einem Benutzer angesprochen werden und liefert eine Information zurück. Dies kann zum Beispiel ein Wetter-Webservice sein (siehe Kapitel 4.4.1), das die aktuelle Temperatur weitergibt oder auch ein Aktien-Webservice, das den aktuellen Aktienkurs für eine Aktie zurückliefert. Im Prinzip kann man sich vorstellen, dass ein Webservice Informationsdaten, die zum Beispiel aus einer Datenbank kommen, an alle anfragenden Benutzer verteilt. Mit Webservices kann man Programmierer sehr gut unterstützen, da man Webservices mit jeder Programmiersprache und jeder Plattform ansprechen und verarbeiten kann. Ein wesentlicher Vorteil ist die Plattformunabhängigkeit[23].

Um noch zusätzlich auf eine wesentliche Erleichterung für Programmierer einzugehen, empfiehlt sich wieder das Wetter-Webservice-Beispiel. Der Webservice-Aufruf für dieses Webservice ist im Programm-Code in etwa drei Zeilen lang. Mithilfe des Programms WSImport, das seit der Version Java Development Kit 1.6 (JDK 1.6) mit ausgeliefert wird, kann man sich sehr einfach Webservice relevante Klassen für die Verwendung eines Webservices generieren lassen. (Oracle JDK) Das Programm kann in jeder Konsole mit dem folgenden Aufruf ausgeführt werden:

```
C:\<Ort der JDK>\wsimport -keep -verbose http://wetter-webservice.at/wetter-webservice?wsdl
```

(SUN WSImport) Der Aufruf erzeugt alle relevanten Java Klassen, um ein Webservice aufrufen zu können. Diese Klassen können in ein Java Projekt eingebunden werden und mithilfe weniger Zeilen kann man das Webservice aufrufen:

```
// Das Webservice-Objekt wird mit Hilfe der Klasse WetterWS_Service initialisiert
WetterWS_Service myWetterWS_Service = new WetterWS_Service();
// Durch die Methode getWetterWSPort kann das Service angesprochen werden
WetterWS myWetterWS = myWetterWS_Service.getWetterWSPort();
// Der Aufruf des Webservices mit den benötigten Attributen
float temperatur = myWetterWS.getWetter("Wien", "Celsius");
```

Locator (URL) oder ein Uniform Resource Name (URN). Die URL ist im Internet die häufigste Verwendung der URI, da man mit Hilfe der URL eine Ressource finden kann, die hinter der URL steht. Eine URL ist immer einzigartig, kann aber auf dieselbe Ressource wie eine andere URL zeigen. (URI Generic Syntax) (Webservices mit REST)

[23] Plattformunabhängigkeit bedeutet, dass man von keinem Betriebssystem und von keiner Systemarchitektur abhängig ist. Ein Web Service läuft auf allen gängigen Betriebssystemen mit jeder Art der Hardware.

Das Webservice wird mit den ersten zwei Zeilen adressiert und die letzte Zeile führt das Webservice mit der Methode „getWetter“ („Wien“, „Celsius“) aus. Das Webservice liefert die aktuelle Temperatur des Ortes „Wien“ in der Einheit „Celsius“ zurück.

Wenn man diese Information nun nicht über ein Webservice zur Verfügung gestellt bekommt, kann man auch anders an die benötigten Daten herankommen. Da diese Methode nicht direkt mit der Lösungsfindung verbunden ist, wird sie im Anhang in Kapitel 13.1 Alternative zu einem Webservice beschrieben.

Im betrieblichen Umfeld werden sehr viele Webservices verwendet, um Informationen und Geschäftsprozesse abzuwickeln. Im Zeitalter der multinationalen Unternehmen müssen Geschäftsprozesse und Unternehmensinformationen über Landesgrenzen hinaus verteilt werden. Um diesen Anforderungen gerecht zu werden, werden die unzähligen Webservices, die in Firmen intern und extern bereitgestellt werden, über die Beschreibungssprache Business Process Execution Language (BPEL)[24] orchestriert[25]. Der Internethändler Amazon bietet sogar Rechenleistungen von seinen riesigen Serverfarmen über Webservices an. Man kann die Rechenleistung und Konfigurationen auf den virtuellen Maschinen per Webservice steuern. (Amazon Web Service 1) (Amazon Web Service 2)

4.4. SOAP vs. REST

Bei der Frage Representational State Transfer (REST) oder Simple Objekt Access Protocol (SOAP) soll geklärt werden, welche der beiden Technologien am geeignetsten ist, um eine Applikation auf einem Smartphone mit Daten zur Darstellung von Modellen zu beladen. Diese beiden unterschiedlichen Arten von „Webservice-Technologien“ können von allen gängigen Smartphonebetriebssystemen eingesetzt werden. Der Informationsaustausch zwischen einem Datenbereitsteller und dem Endgerät eines Benutzers kann mit beiden Technologien durchgeführt werden, es sollte jedoch die schnellere, einfachere, wiederverwendbarere, weiterentwickelbarere und Ressourcen schonendere Technologie für eine Modellerierungsapplikation gewählt werden.

[24] BPEL basiert auf XML und wird zur Beschreibung von Geschäftsprozessen verwendet. Die Geschäftsprozesse werden durch Web Services implementiert. BPEL wurde von der Organization for the Advancement of Structured Information Standards (OASIS), eine ähnliche Organisation wie die W3C, standardisiert.
[25] Bei der Orchestrierung werden mehrere Web Services zu einer Komposition zusammengefasst.

Die Technologie REST bzw. ein RESTful Webservice ist kein Standard, im Gegensatz zum offiziellen Standard SOAP, sondern eine Art Architekturstil. Ein RESTful Webservice wird zumeist verwendet, wenn ein Ressourcenzugriff durchgeführt wird. Der Standard SOAP wird zumeist bei Aufrufen von entfernten Objekten und im Zusammenhang mit deren Zuständen eingesetzt. (Java Insel 7) Im weiterem werden beide Technologien genauer erklärt und im Kapitel 4.4.3 miteinander verglichen.

4.4.1. SOAP

Simple Objekt Access Protocol soll es ermöglichen Daten zwischen Systemen auszutauschen und dies soll mit Hilfe des Protokolls RPC durchgeführt werden. Die Abkürzung SOAP ist erhalten geblieben, auch wenn sich das Protokoll seit Version 1.2 (Recommendation der W3C vom Juni 2003) nicht nur auf Objekte bezieht und nach der Quelle (Einstieg in XML) nicht mehr simple ist. Das Protokoll SOAP stützt sich auf das XML-Format und wird meistens über HTTP übertragen. Mit Hilfe des SOAP-Protokolls können einfach Daten und Informationen ausgetauscht werden. Eine praktische Anwendung von SOAP kann jede Kommunikation mit einem Web Service sein, wie zum Beispiel die Wetter-App auf einem Smartphone oder das Wetter-Widget auf dem Desktop eines Computers. Die Applikation holt die aktuellen Wetterinformationen über ein Webservice ein, sobald man die Applikation startet. Nach dem Start wird eine Anfrage an ein Yahoo! Webservice gestellt und die Applikation zeigt das aktuelle Wetter für die gewünscht Stadt an. Weitere bekannte Beispiele für den Einsatz von SOAP sind ebay und Amazon, die ihre Suchanfragen über das Protokoll SOAP abwickeln. (Einstieg in XML)

Eine SOAP-Message besteht aus mindestens zwei Teilen: SOAP-Envelope (deutsch: Umschlag) und SOAP-Body (deutsch: Körper). Optional kann noch ein SOAP-Header (deutsch: Kopf) hinzugefügt werden, dieser ist aber für eine Kommunikation mit SOAP nicht notwendig. Falls man ein Header Tag in dem SOAP-Envelope mitschickt, muss es als erstes angeführt werden.

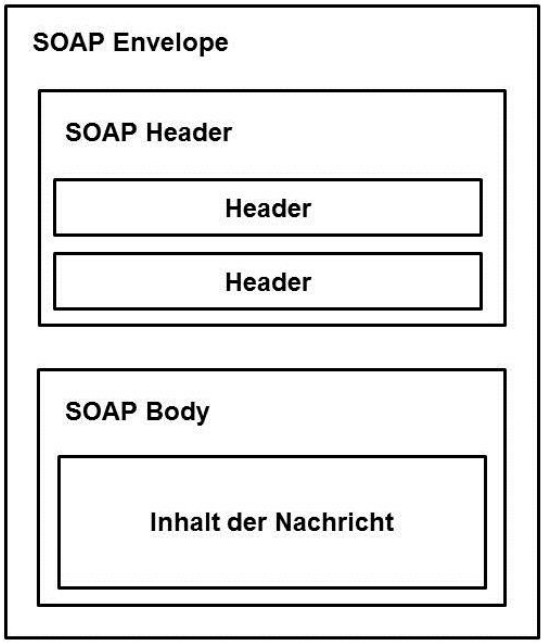


Abbildung 40 zeigt den Aufbau einer SOAP-Nachricht (Zitat von Abbildung 10.11 (Einstieg in XML) Seite 453)

Der SOAP-Envelope umfasst alle nötigen Informationen und man kann ihn sich wie einen Briefumschlag vorstellen. Der Header kann unterschiedliche Daten umfassen, ist aber, um eine valide SOAP-Nachricht zu verschicken, nicht notwendig. Bei den Daten kann es sich um Attribute, Encoding oder auch Passwörter und Usernamen handeln. Im SOAP-Body sind die eigentlichen Daten vorhanden, die der Benutzer angefragt hat. In dem oben genannten Beispiel wären hier die Wetterdaten einer bestimmten Stadt gespeichert oder bei einer Suchanfrage über ein Buch bei Amazon wären die Suchergebnisse der Anfrage im Body gespeichert.

Eines der einfachsten Szenarien ist ein „Anfrage-Antwort-Szenario“ und wird im folgenden Beispiel dargestellt.

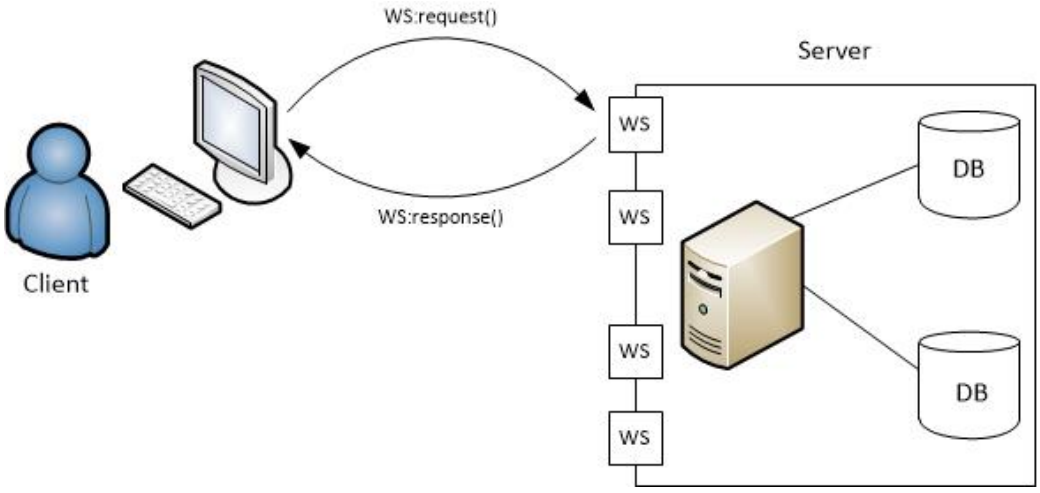


Abbildung 41 zeigt ein Client-Server-Szenario, in dem der Client einen Aufruf (WS:request) an den Server sendet und vom Server eine Antwort (WS:response) bekommt.

Der Benutzer (in Abbildung 41 dargestellt als Client) schickt eine Anfrage an den Server, der unterschiedliche Webservices (in Abbildung 41 dargestellt durch WS) zur Verfügung stellt, um Informationen verteilen zu können. Die Anfrage an den Server könnte vom Client wie folgt aussehen:

```
<?xml version="1.0" encoding='UTF-8'?>
<soap-envelope:Envelope
    xmlns:soap-envelope="http://schemas.xmlsoap.org/soap/envelope/">
  <soap-envelope:Header>
    <transaction:Trans xmlns:transaction="http://www.w3schools.com/transaction/“
        soap-envelop:mustUnderstand="1">test
    </transaction:Trans>
  </soap-envelope:Header>
  <soap-envelope:Body>
    <yahoo-wetter:checkWetter xmlns:yahoo-wetter="http://de.wetter.yahoo.com">
      <yahoo-wetter:Stadt>Wien</yahoo-wetter:Stadt>
    </yahoo-wetter:checkWetter>
  </soap-envelope:Body>
 </soap-envelope:Envelope>
```

Diese Beispiel SOAP-Anfrage soll beim Wetter-Webservice der Seite Yahoo das Wetter für die Stadt Wien erfahren. Hierzu würde man diese SOAP-Nachricht in eine HTTP-Anfrage einbinden und an das Yahoo-Webservice senden. (SOAP Body) (SOAP Header) (SOAP Envelope)

Für dieses Beispiel wurde ein SOAP-Header hinzugefügt, in dem man das Standardattribut „mustUnderstand“ eingebunden hat. Dieses Attribut kann die Werte

„1“ oder „0“ annehmen. Das Attribut kann steuern, ob ein Header verpflichtend („1“) oder optional („0“) ist. (SOAP Body) (SOAP Header) (SOAP Envelope)

Die Antwort vom Yahoo-Webservice könnte wie folgt aussehen:

```
<?xml version="1.0" encoding='UTF-8'?>
<soap-envelope:Envelope xmlns:soap-
  envelope="http://schemas.xmlsoap.org/soap/envelope/">
  <soap-envelope:Header>
    <transaction:Trans xmlns:transaction="http://www.w3schools.com/transaction/" soap-
      envelop:mustUnderstand="1">test
    </transaction:Trans>
  </soap-envelope:Header>
  <soap-envelope:Body>
    <yw: checkWetter xmlns:yw="http://de.wetter.yahoo.com">
      <yw:aktuellesWetter>24,5°</yw:aktuellesWetter>
    </yw:checkWetter>
  </soap-envelope:Body>
</soap-envelope:Envelope>
```

Die Antwort beinhaltet auch wieder den Header, den man durch das Attribut „mustUnderstand“ als notwendig gekennzeichnet hat, und hat im Body die Antwort auf die Anfrage eingebettet. Das Webservice hat die Antwort gesendet, dass es in der Stadt Wien, auf die sich die Anfrage bezogen hat, 24,5° hat. Dies kann am Ausgabegerät des Benutzers angezeigt werden. (SOAP Body) (SOAP Header) (SOAP Envelope)

Damit man ein Webservice mit der Technologie SOAP aufrufen kann, muss man wissen, welche Parameter bei dem Serviceaufruf übergeben werden müssen. Damit man auch fremde Webservices verwenden kann, wird ein Webservice mit der Auszeichnungssprache WSDL beschrieben. In einer WSDL-Datei kann man herauslesen, welche Parameter bei der Anfrage an ein Webservice mitgesendet werden müssen, wie im Kapitel 4.3 erläutert wurde. In dem Beispiel aus Abbildung 41 würde das Webservice ein Tag mit dem Namen „checkWetter“ und ein Tag mit dem Name „Stadt" erwarten. Durch diese Parameter kann das Webservice die gewünschten Informationen in den Tags „wetterAntwort“ und „aktuellesWetter“ zurücksenden. Die WSDL-Datei zeigt auch an, wie die Antwort aussehen wird, daher kann der Client die Antwort des Webservices auswerten. (Einstieg in XML)

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap/
  xmlns:soapenc=http://schemas.xmlsoap.org/soap/encoding/
  xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:yw=http://de.wetter.yahoo.com
  xmlns:http=http://schemas.xmlsoap.org/wsdl/http/
```

```
  targetNamespace=http://www.example.net
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
      targetNamespace="http://de.wetter.yahoo.com">
      <s:element name="checkWetter">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="Stadt" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="checkWetterResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="aktuellesWetter" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="checkWetterSoapIn">
    <wsdl:part name="parameters" element="yw:checkWetter" />
  </wsdl:message>
  <wsdl:message name="checkWetterSoapOut">
    <wsdl:part name="parameters" element="yw:checkWetterResponse" />
  </wsdl:message>
  <wsdl:portType name="checkWetterSoap">
    <wsdl:operation name="checkWetter">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Zeigt das Wetter
        für eine mitgeschickte Stadt an!
      </wsdl:documentation>
      <wsdl:input message="yw:checkWetterSoapIn" />
      <wsdl:output message="yw:checkWetterSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="checkWetterSoap" type="yw:checkWetterSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="checkWetter">
      <soap:operation soapAction="http://de.wetter.yahoo.com/checkWetter"
        style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="checkWetter">
    <wsdl:port name="checkWetterSoap" binding="yw:checkWetterSoap">
      <soap:address location="http://de.wetter.yahoo.com/checkWetter.wsdl" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

In der Beispiel-WSDL des Wetter-Webservices wird eine Methode mit dem Namen „checkWetter“ im message Tag definiert. Das Webservice erwartet, wie im Tag types im XSD zu sehen, einen Städtenamen „Stadt“ vom Datentyp string und liefert eine Antwort „aktuellesWetter“ vom Datentyp string mit dem aktuellen Wetter zurück. Beide Variablen müssen mindestens 1-mal vorkommen und können auch maximal 1-mal vorkommen. In den Tags binding wird unter anderem der Ort des Service, der Input und Output beschrieben. Im service Tag wird u.a. die Adresse der WSDL-Datei hinterlegt. Eine nähere Beschreibung zu den Tags wurde bereits in Kapitel 4.3 erklärt.

Die SOAP-Technologie wurde bereits in vielen unterschiedlichen Programmiersprachen implementiert und wird dort auch aktiv verwendet. Es gibt eine Implementierung für C/C++, Phyton, PHP, Java, .Net und viele andere Programmiersprachen. Die Bibliothek, kSOAP 2, wurde von einigen Benutzer als Java Platform 2, Micro Edition (J2ME)[26] Library[27] entwickelt. Diese Bibliothek kann man auch in Android einbinden. (Sourceforge kSOAP2) Wie diese aussehen kann, wird in Kapitel 4.4.3 gezeigt und näher erklärt.

4.4.2. REST

Representational State Transfer bezeichnet eine mögliche Architektur für Webservices, die sich vor allem auf die vorhandenen Objekte bzw. Ressourcen konzentriert. REST ist kein einheitlicher Standard, so wie SOAP, sondern ein Architekturvorschlag, der bestimmte Grenzen für einen REST-konformen Aufbau bereitstellt. Roy Fielding, Informatiker und einer der Autoren der HTTP-Spezifikation, hat sich in seiner Dissertation mit dem Thema REST und unter anderem auch den Architekturabgrenzungen beschäftigt:

„scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems“ (Zitat (REST) Kapitel 5.5 Seite 105 2. Absatz)

Dieses Zitat sagt aus, dass ein REST-Webservice folgende Eigenschaften aufweisen soll:

[26] J2ME kann als Java für Mobiledevices bezeichnet werden.
[27] Eine Android-Library ist ein Zusammenschluss aus verschiedenen Methoden und Klassen, die der Programmierer für sein eigenes Programm wiederverwenden kann.

- Skalierbarkeit der Interaktionen zwischen Komponenten
- Allgemeingültigkeit von Interfaces
- Unabhängige Entwicklung von Komponenten
- Zwischengeschaltete Komponenten, um die Wartezeit zu verkürzen
- Erzwungene Sicherheit
- Abkapseln von alten Systemen

Ein REST-konformes Webservice muss folgende Eigenschaften und Prinzipien umsetzen:

- Adressierbarkeit: Ein REST-konformes Webservice muss eindeutig anhand einer URL adressierbar sein. Weiters stellt so ein Service oft Ressourcen über URIs zur Verfügung.
- Unterschiedliche Repräsentationen: Ein REST-konformes Webservice kann Daten in benötigten Formen ausgeben (umwandeln) und ist somit sehr stark an den Bedürfnisse der Benutzer orientiert. Das Service kann Daten zum Beispiel in XML oder JSON liefern und der Client kann immer dieselbe URL bzw. URI adressieren.
- Zustandslosigkeit: Ein REST-konformes Webservice setzt auf eine zustandslose Kommunikation zwischen Client und Server. Der REST-Aufruf beinhaltet alle benötigten Informationen, die zur Weiterverarbeitung der Nachricht nötig sind. Das gilt sowohl für Nachrichten an den Server, als auch für Nachrichten an den Client. Die Zustandslosigkeit hat den Vorteil, dass die Anfragen in sich geschlossen sind und keine dauerhafte Kommunikation (ressourcenschonend) zwischen Client und Server notwendig ist. Auch ist es für keine der beiden Seiten notwendig, Informationen zu speichern, um später auf diese wieder zugreifen zu können. Das Übertragungsprotokoll ist meistens HTTP oder HTTPS.
- Operationen: Ein REST-konformes Webservice muss Operationen wie etwa GET, POST, HEAD, PUT, OPTIONS und DELETE zur Verfügung stellen. Diese Operationen sind, sowie im HTTP-Standard beschrieben, zu implementieren. Auf diese Operationen wird im nächsten Absatz noch näher eingegangen.
- Verwendung von Hypermedien: Ein REST-konformes Webservice soll dem Benutzer ermöglichen von Ressourcen weiter navigieren zu können, ohne dass es zu weiteren Ladezeiten oder Client-Server-Aufrufen kommt.

(REST) Die oben genannten Operationen sollen folgende Funktionen abdecken:

- GET: Fordert eine spezielle Ressource vom Server an. Am Server kann nichts verändert werden, daher ist die GET-Methode als sehr sicher einzustufen.
- POST: Fügt eine untergeordnete Ressource am Server hinzu.
- HEAD: Hat dieselbe Funktion und Sicherheit wie die Methode GET, nur dass es keinen Message Body in der Antwort gibt. Hier werden Meta-Daten zu einer Ressource zurückgeschickt.
- PUT: Legt eine Ressource unter einer speziellen URI ab. Falls dort bereits eine Ressource vorhanden ist, wird diese aktualisiert.
- OPTIONS: Gibt eine Liste an Möglichkeiten und Operationen für diese Ressource zurück.
- DELETE: Löscht die Ressource, die hinter der URI liegt. Zumeist wird die Ressource jedoch nicht komplett gelöscht, sondern deaktiviert oder löschmarkiert (versteckt).

(REST) Diese Operationen decken alle notwendigen Funktionalitäten ab, um Ressourcen zu bearbeiten oder abzufragen. Mehr Funktionen sind in der REST-Welt auch nicht notwendig, da man nur hier die Zustandslosigkeit, die der Service haben soll, auch garantieren kann. Der Client muss sich auch gegebenenfalls beim Server bei jedem Request neu authentifizieren, da die Kommunikation zwischen diesen zustandslos abläuft und der Server nach der Abarbeitung des Requests den Client schon wieder „vergessen" hat. Eine typische Kommunikation zwischen einem Client und einem Server könnte mit einem REST-Webservice, wie in folgender Abbildung dargestellt, aussehen:

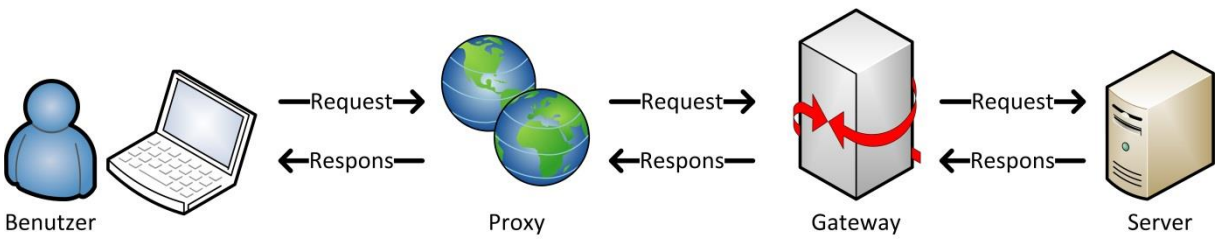


Abbildung 42 zeigt einen Benutzer, der einen Request über einen Proxy und einen Gateway an einen Server schickt. Dieser Benutzer erhält vom Server über den rückwertigen Weg einen Respons.

Jede Station hat ihre eigene Aufgabe und übermittelt die Informationen für die nächstfolgende Station. Es wird nur mit dem unmittelbaren Nachfolger interagiert und, nachdem die Aufgabe erledigt wurde, wird der vorherige Zustand wiederhergestellt, da die Kommunikation zustandslos in beide Richtungen abläuft. Auf der Seite des Clients kann es jedoch Sinn machen, wenn man häufige Anfragen im Cache abspeichert, um nicht immer alle Informationen hin- und hersenden zu müssen. Der Cache könnte auch die Datenpacketgröße erheblich verringern, was zu schnelleren Aufrufen und Antworten führen kann und zusätzlich ressourcenschonender für die Seite des Servers ist. Es ist nur sinnvoll Daten zu cachen, die sich nicht allzu oft ändern. Datum und Uhrzeiten von Webseiten zu cachen macht keinen Sinn, jedoch Logos und Banner einer Homepage kann man durchaus im Cache zwischenspeichern, da sich diese nicht oft ändern werden. Hier muss eine Information vom Server mitgesendet werden, welche Informationen sinnvoller Weise im Cache behalten werden sollen und welche verworfen werden können. (REST) (Webservices mit REST)

REST kann durch die Plattformunabhängigkeit und die Programmiersprachenunabhängigkeit in jeglicher erdenklicher Kombination umgesetzt werden. (Java Insel 7)

4.4.3. Vergleich

Der Vergleich von SOAP und REST ist nicht allzu einfach, da REST ein Architekturstandard für ein Webservice ist, und SOAP ein standardisiertes Protokoll zur Datenübertragung darstellt. Man kann beide Techniken im Falle dieser Arbeit auf ein gemeinsames Level bringen, um einen sinnvollen Vergleich anstellen zu können. Beide Technologien können sowohl HTTP als auch die sicherere Variante HTTPS verwenden. Sowohl REST als auch SOAP können Cachespeicher einsetzen, um die unnötige Belastung vom Netzwerk bzw. Belastung vom Server möglichst gering zu halten.

Der Weg beider Technologien führt mittels HTTP(S) zum Ziel. Jedoch verwendet hier SOAP meistens die POST-Methode und REST die GET-Methode von HTTP. SOAP verpackt alle notwendigen Parameter und Daten für einen Methodenaufruf in einem SOAP-Envelope, der wiederrum in einem HTTP-Body verpackt wird. Diese Nachricht wird mit der POST-Methode via HTTP an den Server übermittelt. Im Gegensatz dazu verwendet ein REST-konformes Webservice direkt die HTTP-Methode GET, um die

Parameter an einen Server zu übermitteln. Diese werden einfach bei Methodenaufruf als Parameter mitgegeben und können aus der URL abgelesen werden. Bei einfachen Abfragen ist hier der REST-Service um einiges einfacher zu verwenden und auch schneller als der SOAP-Service. Man kann Webservices direkt adressieren und Parameter mithilfe der URL an den Server übermitteln. Anbei ein Beispielcode, wie ein solcher Aufruf für das Beispiel des Wetter-Webservices aussehen könnte:

	SOAP	**REST**
Methode:	POST	GET
URL:	http://de.wetter.yahoo.com/search/	http://de.wetter.yahoo.com/search?q=REST ?start=0?maxResults=10?city=vienna
Version:	HTTP/1.1	HTTP/1.1
Inhalt:	<Envelope> <Body> <doYahooSearchWeather> <q>SOAP</q> <start>0</start> <maxResults>10</maxResults> <city>Vienna</city> </doYahooSearchWeather> </Body> </Envelope>	

Tabelle 4 zeigt den Vergleich zwischen einem SOAP-Aufruf (links) und einem REST-Aufruf (rechts) vom aktuellen Wetter in der Stadt Wien durch das yahoo.com-Wetter-Webservice

Die Quelle (Java Insel 7) fasst den Unterschied zwischen beiden Technologien sehr gut zusammen: „Wenn es Ressourcen-Zugriffe gibt und es keine entfernten Aufrufe an wirkliche Objekte mit Zuständen und Verhalten gibt, ist ein RESTful Web-Service eine gute Wahl. Stehen entfernte Objekte mit ihren vielfältigen Funktionen im Vordergrund, ist ein SOAP Web-Service in Ordnung.“ (Zitat aus (Java Insel 7) Seite 1058 Absatz 8) Da es sich hier um gespeicherte Modelle handelt, die sich immer wieder verändern können, und das Hauptaugenmerk auf Geschwindigkeit der Datenbeschaffung gelegt wird, sollte man hier zu einem RESTful Webservice tendieren.

Weiters sei noch anzumerken, dass iOS, Windows Phone und auch Android REST Webservice direkt aufrufen können und keine weitere Implementierung von Klassen oder Codebibliotheken notwendig ist. Bei SOAP muss man sich Programmbibliotheken von Drittanbietern herunterladen und extra in die App einbinden, um diese SOAP-

Services verwenden zu können. Bei Android heißt diese ksoap und bei iOS gibt es die Bibliothek csoap. Diese werden aber von privaten Entwicklern bereitgestellt und man kann sich nie sicher sein, ob diese up-to-date sind. Durch die Wartung von privaten Entwicklern kann man auch nicht immer darauf vertrauen, dass diese Programmbibliotheken in regelmäßigen Abständen Sicherheitsupdates durchlaufen.

Es folgt der Aufruf eines vereinfachten Wetter-Webservices. Zur Veranschaulichung und zum besseren Überblick wurden die Antworten der Webservices ein wenig gekürzt. Die originalen Dateien können in der Quelle (CDYNE WetterWS) eingesehen werden.

Ein Webservice-Aufruf mit SOAP unter Android:

```
public static SoapObject createRequest(String zipCode) {
  SoapObject soaprequest = new SoapObject("http://ws.cdyne.com/WeatherWS/",
"GetCityForecastByZIP");
  PropertyInfo property = new PropertyInfo();
  property.setNamespace("http://ws.cdyne.com/WeatherWS/");
  property.setName("ZIP");
  property.setValue("1230");
  request.addProperty(property);
  return request;
}

@Override
protected CityForecastBO performTaskInBackground(SoapObject parameter) throws
Exception {
  SoapObject soapparameter = getSOAPRequestParameter();
  SoapSerializationEnvelope soapenvelope = new
SoapSerializationEnvelope(SoapEnvelope.VER11);
  soapenvelope.setOutputSoapObject(soapparameter);

  HttpTransportSE httpTransport = new
HttpTransportSE("http://wsf.cdyne.com/WeatherWS/Weather.asmx?WSDL");

  httpTransport.call("http://ws.cdyne.com/WeatherWS/" + "GetCityForecastByZIP",
soapenvelope);

  SoapObject soapResultObject = (SoapObject) soapenvelope.bodyIn;
  result = parseSOAPResponse(soapResultObject);
  return result;
}

private String parseSOAPResponse(SoapObject response) {
  String parseresult = "";
  SoapObject prognose = (SoapObject) response.getProperty("GetStadtbyPLZ");
  String stadt =  prognose.getPrimitivePropertySafelyAsString("Stadt");
  String temperatur =  prognose.getPrimitivePropertySafelyAsString("Temperatur");
  parseresult = "In der Stadt: "+stadt+" hat es "+temperatur"+" Grad.";
  return parseresult;
}
```

(ksoap Beispiel) Mit Hilfe der Programmbibliothek ksoap kann man einige Funktionen bereits nutzen ohne diese implementieren zu müssen. In diesem Programmbeispiel wird zuerst ein „SoapObject“ mit den nötigen Parametern erzeugt, wie etwa dem Namespace der Postleitzahl und dem Wert für die Postleitzahl (in diesem Fall war es die Postleitzahl 1230). Danach wird zuerst ein SOAP-Envelope erzeugt mit dem „SoapObject“, das die vorhin erwähnten Parameter besitzt. Danach wird eine HTTP-Connection erzeugt und das Webservice wird mit dem SOAP-Envelope bestückt aufgerufen. Dies sendet dann eine Antwort zurück, die im „soapResultObject“ gespeichert wird. In der Methode „parseSOAPResponse“ werden die Ergebnisse des SOAP-Bodys ausgelesen. Die Antwort könnte in etwa so aussehen:

```
<soap:Body>
  <GetStadtbyPLZ xmlns="http://ws.cdyne.com/WeatherWS/">
    <Stadt>Wien</Stadt>
    <Temperatur>20</Temperatur>
  </GetStadtbyPLZ>
</soap:Body>
```

Da man aus der WSDL-Datei die erwartete Rückgabestruktur auslesen kann, kann man mit Hilfe der ksoap-Bibliothek genau auf die gewünschten Elemente zugreifen.

Ein Webservice-Aufruf mit REST unter Android:

```
public class Rest {
  private int timeout = 30000;
  private HttpClient httpClient;
  private HttpPost httpPost;
  private HttpGet httpGet;
  private HttpResponse httpResponse;
  private String responseString = null;
  private ArrayList data;

  public Rest(){
    this.httpClient = new DefaultHttpClient();
    this.data = new ArrayList();
  }

  public void get(String url){
    this.httpGet = new HttpGet(url);
    this.httpClient.getParams().setParameter("http.socket.timeout", this.timeout);
    try { this.httpResponse = this.httpClient.execute(this.httpGet); }
    catch (Exception e) {}
  }
```

```
  public String getResponseString(){
    if(this.httpResponse!=null){
      try {
        HttpEntity httpEntity = this.httpResponse.getEntity();
        InputStream is = httpEntity.getContent();
        BufferedReader reader = new BufferedReader(new InputStreamReader(is, "utf-8"), 8);
        StringBuilder sb = new StringBuilder();
        String line = null;
        while ((line = reader.readLine()) != null) {
          sb.append(line + "\n");
        }
        is.close();
        this.responseString = sb.toString();
        return this.responseString;
      } catch (Exception e) {}
    }
    return null;
  }
}
```

Der Code zeigt ein einfaches Webservice, das nach dem REST-Prinzip aufgebaut ist. Dieses Webservice kann von einem Benutzer mit folgendem Code aufgerufen werden:

```
Rest r = new Rest();
r.get("http://api.search.yahoo.com/WebSearchService/V1/webSearch?appid=YahooWeather
&results=10");[28]
String textRetrieved = r.getResponseString();
```

Um das Service zu nutzen, muss zuerst ein „Rest“-Objekt initialisiert werden. Danach kann das Objekt mit der Methode „get“ den Konnex zu dem Webservice herstellen. Nachdem diese Verbindung aufgebaut wurde, kann man mit der Methode „getResponseString“ den vom Server bereitgestellten Inhalt abfragen. Das Webservice liefert den gesuchten „String“ an den Benutzer zurück und das Endgerät kann die Informationen für den Benutzer aufbereiten. (REST) (Webservices mit REST) (Yahoo Weather WS)

SOAP Nachrichten müssen immer eine valide XML-Syntax besitzen. Dies ist bei REST nicht der Fall, da REST unter anderem mit JSON arbeiten kann. Diese zwei Technologien werden in dem folgenden Kapitel genauer erläutert und am Ende des Kapitels miteinander verglichen.

[28] Dieses Service wurde von yahoo mittlerweile deaktiviert und durch die neue Yahoo Query Language (YQL) ersetzt. Diese Query Sprache kann mithilfe einer URL und einem SQL (Structured Query Language ist eine Abfragesprache für Datenbanken) ähnlichem Statement Daten von Yahoo abfragen. Dieser Webservice wird als RSS (Really Simple Syndication siehe Kapitel 4.5.2) Feed angeboten und liefert eine XML Datei zurück. (Yahoo WS Update) (Stand 21.03.2014)

4.5. Datenserialisierung

Bei der Datenerialisierung werden Daten in eine Sequenz von Zahlen umgewandelt, damit diese einfacher versendet werden können. Man muss diese Zahlen auch wieder in Daten umwandeln können, dieser Vorgang wird auch Deserialisierung genannt. Eine der einfachsten Umwandlungsformen ist die ASCII-Kodierung. Diese Kodierung teilt jedem Buchstaben, Sonderzeichen und jeder Zahl eine Zahl zwischen 0 und 255 zu, um einen Text einfach in eine Zahlensequenz umzuwandeln. Diese Zahlenfolgen kann man sehr leicht mit Hilfe eines Texteditors wieder in einen Text transformieren. Dies kann man mit binär serialisierten Daten nicht machen, dafür kann eine binär serialisierte Datei platzsparender sein. Die binäre Serialisierung wird häufig bei Bilder-, Audi- oder Videodateien durchgeführt. Um diese wieder lesen zu können, ist ein spezielles Programm notwendig und kein einfacher Texteditor.

Es gibt nun die Möglichkeit, dass Daten für Modelle serialisiert oder binär serialisert an Smartphones versenden werden. Hierfür gibt es unterschiedlichste Methoden und Übertragungsstandards:

	Standardisiert	Lesbar für den Menschen	Binär	Schema
JSON	Ja	Ja	Nein	Teilweise
XML	Ja	Ja	Teilweise	Ja
BSON	Ja	Nein	Ja	Nein
Apache Avro	Ja	Nein	Ja	Ja
Apache Thrift	Nein	Teilweise	Ja	Ja
Bencode	Ja	Teilweise	Teilweise	Nein

Tabelle 5 zeigt einige Serialisierungsverfahren und deren Eigenschaften

Die in Tabelle 5 gezeigten Serialisierungsverfahren sind nicht alle, die es gibt, aber eine Auswahl aus den gängigsten binären und nicht-binären Verfahren. BSON steht für „Binary JSON“ und baut auf der JSON-Technologie auf. Bei XML gibt es einen Standard namens Binary-XML, der binäre XML-Daten verarbeitet. Dies soll den Vorteil haben, dass die Daten weniger Speicherplatz verbrauchen. (Galileo Serialisierung)

Ein wesentlicher Punkt bei diesen Verfahren ist die Lesbarkeit für den Menschen. Viele Formate können Binärdaten senden, empfangen und serisaliseren bzw. deserialisieren,

aber diese Daten sind für den Menschen unlesbar. Damit die Lesbarkeit der Modelldaten, die versendet und empfangen werden, gewährleistet wird und da aktuell die Smartphone Betriebssysteme besser mit JSON und XML umgehen können, konzentriert sich diese Arbeit auf diese zwei Formate. In Zukunft kann es durch aus sein, dass binäre Serialisierungsverfahren stärker in die mobilen Betriebssysteme mit eingebunden werden.

In der Programmiersprache des Betriebssystems Android wird sehr stark auf Extensible Markup Language (XML) vertraut. Ein Großteil des Designs der Applikation und des Interfaces wird in XML definiert und geschrieben. Dies ist ein erheblicher Vorteil, da man die Benutzerinterfaces dadurch sehr gut ordnen und auch steuern kann. Es gibt einen wesentlichen Vorteil der Wiederverwendbarkeit beim Nutzen von XML in diesem Zusammenhang. In diesem Kapitel soll eine alternative Technologie aufgezeigt werden, mit der man zwar die Interfaceprogrammierung nicht ersetzen kann, aber dafür die Datenübertragung. Aktuell wird in sehr vielen Quellen auf die einfache Verwendung von XML im Datenaustausch hingewiesen, weil die Umsetzung im Programm wirklich sehr einfach ist und schnell funktioniert. (Einstieg in XML) (Java Insel 7) Manche Quellen weisen aber auch darauf hin, dass dieser Austausch mit der JavaScript Object Notation (JSON) schneller sein kann und nicht so große Datenpakete mit sich bringt. (Android 2) (Java Web Services) (Java Insel 7)

4.5.1. JavaScript Object Notation

JSON wird als kompaktes Datenformat angesehen, das sowohl von Menschen als auch von Maschinen gelesen werden kann. Diese Notation dient zum Datenaustausch zwischen Anwendungen und wurde von JavaScript ausgehend entwickelt. Douglas Crockford, der diese Notation erstmals in einem Standard spezifiziert[29] hat, betitelt JSON auch als Untermenge von JavaScript. (JSON Fat Free) JSON soll dem Benutzer helfen, einfacher und besser mit dem Datenaustausch zu Recht zu kommen, als es andere Technologien anbieten. Die Notation ist vollkommen Sprachen und Plattform unabhängig und kann daher einfach eingesetzt werden. Durch die Unterstützung von Unicode ist es möglich in allen menschlichen Sprachen Informationen zu transportieren. JSON kann unter anderem mit folgenden Programmiersprachen zum Einsatz kommen:

[29] Der Standard ist auf http://tools.ietf.org/html/rfc4627 zu finden. Dies ist die Homepage der „Internet Engineering Task Force“ und der Entwurf wurde dort im Juli 2006 erstmals eingereicht.

- C / C++
- Java
- Perl
- PHP
- Python
- Ruby

  (eine vollständige Liste kann unter (JSON Fat Free) Absatz 6 eingesehen werden)

Der Autor des Standards verspricht, dass JSON von Mensch und Maschine einfach zu lesen ist, und dass es für Maschinen einfach zu parsen[30] ist. (JSON Fat Free)

Das Format JSON kann folgende unterschiedliche Werte als Inhalt bekommen:

- String: Ein „String“ ist eine Zeichenkette, bestehend aus 0 oder mehr Unicode characters. Es können unterschiedliche Buchstaben, Zahlen oder Zeichen verwendet werden. Der String muss zwischen zwei "" geschrieben werden und darf die folgenden Zeichen nicht in dieser Form enthalten: " oder \ . Man kann verschiedene Zeichen mit einem vorangestellten Backslash verwenden, wie etwa Anführungszeichen, einen Slash oder auch einen Backslash. Zulässig wäre folgender String: "Dies ist ein \"Test-String\"." Der Praser würde als Ergebnis dann Folgendes anschreiben: Dies ist ein "Test-String". Um einen Backslash oder einen Slash darzustellen, würde man Folgendes schreiben: "Ein Backslash \\ und ein Slash \/". Dies würde vom Parser wie folgt ausgelesen werden: Ein Backslash \ und ein Slash /

  Man kann den String auch mit unterschiedlichen Zeichen formatieren:

  - \b (Backspace): Hat dieselbe Funktion wie auf der Tastatur: die letzte geschriebene Position wird gelöscht.
  - \f (Formfeed): Wird im ACSII als „Control Character“ verwendet und hat den Nutzen, dass ein Drucker die aktuelle Seite auswirft und mit einer neuen Seite beginnt. Es wird aber auch zum Beispiel in der Programmiersprache Python verwendet, um den Sprung auf eine neue Seite darzustellen.

[30] Der Ausdruck „to parse“ stammt aus dem englischen und bedeutet so viel wie „etwas zergliedern“ oder auch „etwas analysieren“. „Parsen“ wird verwendet, wenn ein Programm einen Text nach einem gewissen Term oder Wort durchsucht.

    - \n (Newline): Es wird in einer neuen Zeile weitergearbeitet.
    - \r (Carriage return): Siehe \n
    - \t (Horizontal tab): Hat dieselbe Funktion wie der Tabulator auf der Tastatur: erzeugt eine Einrückung nach rechts.
- Number: Als „Number“ kann jede Integer-Zahl, Reelle-Zahl und eine Zahl in wissenschaftlicher Notation verwendet werden. Integer ist eine ganze Zahl zwischen 0 und 65535 oder -32768 und 32767. Eine Reelle Zahl ist jede darstellbare positive und negative Zahl inklusive Kommastellen. Eine Zahl in der wissenschaftlichen Notation wird aufgeteilt in Mantisse und Exponent. Dies wird folgendermaßen dargestellt: $3{,}5*10^3 = 35.000$
- Boolean: Dies heißt, dass Wert „true“ (wahr) oder „false“ (falsch) annehmen kann.
- Object: In JSON sind Objects unsortierte Container mit Paaren von Schlüsseln und Werten. Ein Schlüssel wird vor einen Wert geschrieben, um diesen identifizieren zu können. Ein Schlüssel ist immer ein String und der Wert kann alle Werte annehmen, die in JSON möglich sind. Dies könnte zum Beispiel so aussehen: „name“ : „Test-Name“ In diesem Beispiel ist „name“ der Schlüssel und der String hinter dem „ : “ ist „Test-Name“. Man muss sich an die „normalen“ Regeln der JSON-Werte halten, die in diesem Absatz erwähnt wurden. Ansonsten kann der Wert auch ein anderes Object sein oder auch ein Array. Man kann ein Object mit beliebig vielen Paaren ausstatten oder auch leer lassen. Ein Object muss in JSON mit folgen Klammern begonnen und wieder geschlossen werden: { }
- Array: Ein Array ist eine Folge von Werten. Die Werte im Array können Strings, Numbers, Booleans, Objects oder auch Arrays sein. Ein Array wird mit „ [ ] “ angezeigt und die Elemente im Array werden mit einem „ , “ voneinander getrennt. Ein Array kann aus beliebig vielen Elementen oder auch keinem Element bestehen. Ein Array könnte zum Beispiel so aussehen: [„Jänner“, „Februar“, „März“, „April“, „Mai“, „Juni“, „Juli“, „August“, „September“, „Oktober“, „November“, „Dezember“] Dies wäre ein Beispiel für alle Monate eines Jahres als Array. Die Werte sind jeweils ein gültiger String. Man könnte aber auch ein Array mit Arrays befüllen, wie folgendes Beispiel zeigt: [[1,2,3], [4,5,6], [7,8,9]] Dies

wäre ein Array mit validen Arrays als Elementen, die einfache Zahlen als Werte enthalten.

- null: Der Wert kann „null“ annehmen.

In einer offiziellen Präsentation ist Douglas Crockford kurz auf einige „Regeln“ in der Verarbeitung von JSON eingegangen:

- „Ein JSON Decoder muss alle „well-formed[31]“ JSON Texte akzeptieren“ (Übersetztes Zitat (JSON Fat Free) Absatz 18)
- „Ein JSON Decoder kann auch „nicht-JSON Texte“ akzeptieren“ (Übersetztes Zitat (JSON Fat Free) Absatz 18)
- „Ein JSON Encoder muss „well-formed“ JSON Texte produzieren (Übersetztes Zitat (JSON Fat Free) Absatz 18)

Der Autor empfiehlt, dass „man konservativ sein soll in dem, was man tut, aber liberal in dem, was man von anderen bekommt“. (Übersetztes Zitat (JSON Fat Free) Absatz 18)

Es gibt aber auch einige Argumente bzw. Überlegungen gegen JSON:

- JSON verwendet keine Namespaces: Es ist nicht notwendig auf Namespaces zurückzugreifen, da jedes Object sein „eigener Namespace“ ist und durch die unterschiedlichen Schlüssel alles eindeutig ist.
- JSON hat keinen Validator: Jede Applikation ist selbstverantwortlich für die Validierung des Inputs, daher ist es nicht zwingend notwendig einen eigenen Validator zur Verfügung zu stellen. Weiters sei anzumerken, dass ein YAML[32]-Validator verwendet werden kann.
- JSON ist nicht „extensible“ (erweiterbar): Hier ist „extensible“ in Bezug auf XML zu verstehen. In der Quelle (JSON Fat Free) wird darauf hingewiesen, dass JSON nicht wiederkehrende Datenstrukturen darstellen kann. Weiters ist JSON sehr flexible und kann neue Felder an existierende Strukturen hinzufügen.

[31] „well-formed“ wird übersetzt mit „wohlgeformt“ und bedeutet, dass ein Text den „Regeln“ und der „Grammatik“ einer Programmiersprache bzw. einer Notation entspricht.

[32] YAML steht für „YAML Ain’t Markup Language“ (ursprünglich auch „Yet Another Markup Language“) und ist eine Auszeichnungssprache zur Datenserialisierung. YAML wurde ursprünglich an XML angelehnt, ist aber für den Menschen wesentlich einfacher zu lesen. Bei den Datenstrukturen hat man sich Perl, Python und C als Vorbild genommen. Die Spezifikation wurde von Clark Evans, Brian Ingerson und Oren Ben-Kiki geschrieben.

(JSON Fat Free) Im Großen und Ganzen sei aber festgehalten, dass JSON ein eigener Weg ist, Daten zu übertragen, und es wird versucht, dies möglichst schnell und einfach durchzuführen. Es gibt viele positive Aspekte und einige Einsatzgebiete, in denen man mit JSON mehr bieten kann als beispielweise mit XML. Das hängt jedoch immer vom jeweiligen Einsatzgebiet ab. (JSON XML)

4.5.2. Extensible Markup Language

XML steht für „Extensible Markup Language“ und ist eine so genannte Auszeichnungssprache[33], die dazu dient, Daten in hierarchisch strukturierter Form darzustellen. XML wird vor allem für den Datenaustausch zwischen Computern eingesetzt. Die Sprache ist sowohl für den Menschen als auch für den Computer „lesbar“. Die Auszeichnungssprache wurde zum ersten Mal im Jahr 1998 vom W3C als „Recommendation“ veröffentlicht und ist heutzutage in der fünften Version (die aus dem Jahr 2008 stammt) auf der Homepage des W3Cs zu finden. Diese Spezifikation (W3C XML) beschreibt eine Metasprache, mithilfe derer schon einige „XML-Sprachen“ definiert wurden. Als Beispiele können folgende Sprachen angeführt werden: RSS[34], XHTML[35], XAML[36], SVG[37] und viele mehr. Einschränkungen in den XML-Sprachen können mit Hilfe von Dokumenttypdefinition (DTD) und XML Schema durchgeführt bzw. definiert werden. Ein XML-File wird mit der Datenendung „.xml“ beschrieben.

Ein XML-File muss wohlgeformt sein, damit es einwandfrei verwendet werden kann. Es gibt im Internet freiverfügbare Validators (W3C Validator), um XML-Files online zu überprüfen, oder man kann sich von Altova oder eclipse einen XML-Validator herunterladen und lokal auf einem Rechner installieren.

Folgende Regeln müssen beim Erstellen eines XML-Dokuments beachtet werden:

[33] (englisch: Markup Language) soll den Inhalt eines Dokumentenformats beschreiben. Es werden sogenannte "Tags" gesetzt, an denen sich der Interpreter orientieren kann, um den Inhalt sinnvoll wiederzugeben.
[34] Really Simple Syndication (RSS) ist seit 2000 ein ständig weiterentwickeltes Format, um Änderungen auf Homepages im Internet zu verbreiten. RSS ist in einem XML Format definiert worden.
[35] Extensible HyperText Markup Language (XHTML) ist eine Recommendation der W3C und die aktuelle Version ist XHTML 1.1. Auf die Weiterentwicklung wurde verzichtet, da man HTML5 vorantreiben wollte. XHTML ist eine Auszeichnungssprache, um Bilder und Texte in Dokumenten strukturiert wiederzugeben.
[36] Extensible Application Markup Language (XAML) ist eine Erweiterung von XML und wurde von dem Softwareriesen Microsoft entwickelt. Es dient zur Beschreibung von Oberflächen von Applikationen und Anwendungen.
[37] Scalable Vector Graphics (SVG) ist eine Recommendation der W3C zur Beschreibung von zweidimensionalen Vektorgraphen. Die Sprache SVG basiert auf der XML Notation und ist seit 2008 in der Version 1.2 als Recommendation auf der Homepage des W3C zu finden.

- XML ist eine Auszeichnungssprache und jedes Element muss von einem sogenannten „Tag“ umschlossen sein. Der Beginn eines Elements wird mit „ <> “ dargestellt und das Ende mit „ </> “. In die spitzen Klammern muss eine Beschreibung für das Element eingetragen werden: <Test></Test>. Diese Beschreibung, auch Bezeichner genannt (Einstieg in XML), ist frei definierbar und darf nur mit einem Buchstaben oder Unterstrich beginnen. Weiters können beliebig viele gültige Zeichen (Buchstaben, Zahlen, Unterstriche) verwendet werden. Alle anderen Zeichen, wie etwa Sonderzeichen oder „Sprachenspezifische Zeichen“ wie im Deutschen ä, ö, ü oder ß, sind auch verboten. Man dürfte zwar einen „ : “ verwenden, jedoch ist dieser für Namespaces[38] reserviert. Ein Element kann aber auch so dargestellt werden: <Test/>
- Jedes XML-File muss ein „Root-Element“ (Wurzel-Element) besitzen und dieses muss eindeutig (einzigartig) in diesem Dokument sein.
- Jedes Element kann Attribute besitzen. Diese werden in den Tags definiert: <Test name=“Test-Attribut“/> Jedes dieser Attribute muss einzigartig für dieses Element sein. Das heißt, jedes Element kann in dem vorherigen Beispiel nur ein Attribut mit dem Namen „name“ haben. Jedoch könnte ein anderes Element wie zum Beispiel das Element <Test2 name=“Weiteres-Test-Attribut“/> auch ein Attribut mit dem Namen „name“ haben.
- Ein XML-Dokument muss hierarchisch strukturiert werden. Das heißt, dass sich ein untergeordnetes Element immer komplett in einem über diesem stehenden Element befinden (anfangen und enden) muss. Das Root-Element ist von dieser Regel ausgenommen. Im nächsten Punkt wird dazu ein Beispiel präsentiert.
- Ein zulässiges Beispiel würde wie folgt aussehen:

[38] (deutsch: Namensraum) werden vergeben, um Objekte in einer Baumstruktur eindeutig ansprechen zu können, auch wenn diese denselben Namen haben.

```
<WurzelElement>
  <Ebene1>
    <Ebene2>
      <Ebene3>
      </Ebene3>
    </Ebene2>
    <Ebene2>
    </Ebene2>
  </Ebene1>
</WurzelElement>
```

In diesem Beispiel werden alle Elemente, die in einem Element geöffnet werden, in diesem auch wieder geschlossen. Die hierarchische Struktur wurde eingehalten und es handelt sich um ein wohlgeformtes XML-File.
Ein unzulässiges Beispiel könnte so aussehen:

```
<WurzelElement>
   <Ebene1>
      <Ebene2>
         <Ebene3>
      </Ebene2>
      </Ebene3>
   </Ebene1>
</WurzelElement>
```

Das Element <Ebene3> wird im Element <Ebene2> geöffnet, aber nicht wieder geschlossen. Das schließende Tag befindet sich erst wieder in <Ebene1>. Dies wäre hierarchisch nicht korrekt und es würde beim Validieren zu einem Fehler kommen.

- Ein Element kann folgendes enthalten:
    - Andere Elemente
    - Zeichen: Hier sei anzumerken, dass jedes Zeichen verwendet werden kann. Man muss jedoch beachten, dass manche Zeichen nur mit dem richtigen „encoding[39]“ angezeigt werden können. Andere Zeichen wiederum müssen „umschrieben“ werden, da diese teilweise von einem XML-Parser interpretiert werden würden, ohne Zeichenreferenz:

[39] Encoding bei XML: Es wird dem XML-Dokument ein Standardcode mitgegeben, um dem Interpreter zu zeigen, wie der Text zu lesen ist. Das Encoding hilft dabei, wie man Zeichen lesen und interpretieren muss.

Zeichen	XML-Umschreibung
&	&amp;
<	&lt;
>	&gt;
'	&apos;
»	&quot;

Tabelle 6 zeigt die unterschiedlichen Zeichen, die in einem XML-File umschrieben werden müssen.

- Zeichenreferenzen / Entity-Referenzen: Stellt ein Zeichen wie etwa „&“, „>“ oder auch „<“ mit folgenden Codes dar:
  & = &amp
  < = &lt
  > = &gt
- Verarbeitungsanweisungen: Eine Verarbeitungsanweisung ist eine Anweisung, die von einer auslesenden Software ausgeführt werden soll. Es wird mit zwei „?“, eines nach jeder Klammer, gekennzeichnet:
  <?WetterService alertBox(„es ist eine ungültige Postleitzahl angegeben worden“)?>
  Der Name nach dem „?“ ist als Platzhalter für die auslesende Software gedacht, damit diese erkennt, wann etwas ausgeführt werden soll.
- Kommentare: Ein Kommentar wird von einem Parser ignoriert und sieht wie folgt aus:
  <!-- Kommentar -->
- CDATA-Abschnitte: Werden verwendet um lange Texte im XML einzugeben, ohne auf spezielle Zeichen wie etwa „&“, „<“ oder „>“ achten zu müssen. Innerhalb dieses Tags:
  <![CDATA[Die Zeichen & <> werden vom Parser ignoriert]]>
  kann man beliebigen Inhalt schreiben, ohne dass dieser vom Parser beachtet wird. Ausgenommen ist nur die Kombination „]]>“, da diese das Element schließt und so auch vom Parser interpretiert wird.

- XML ist Case-Sensitive. Daher müssen alle Tags genauso geschlossen werden, wie sie aufgemacht wurden. Zum Beispiel: <Test> </Test> wäre richtig, da beide Bezeichner gleich geschrieben wurden. <TEST> </Test> wäre ein falsch, da „Test“ nicht dasselbe ist wie „TEST“.

- Durch die Zeichenfolge <!-- --> wird in XML ein Kommentar definiert. Ein Parser wird nie versuchen ein Kommentar zu verarbeiten, also können in diesem auch jede Art von Zeichen verwendet werden.

(Einstieg in XML) Man sieht, dass die XML-Sprache einige Vorschriften und Regeln beinhaltet, um ein valides Dokument zu akzeptieren. Im folgenden Kapitel werden die zwei Technologien aus den Kapiteln 4.5.1 und 4.5.2 miteinander verglichen und es folgen Beispiele wie die Umsetzung in Android funktioniert.

4.5.3. Vergleich

Für den Vergleich zwischen den Technologiestandards XML und JSON werden zwei Dateien, eine XML-Datei und eine JSON-Datei, einander gegenübergestellt.

XML-Datei (763 Bytes):

```
<Wetterservice>
<Wetter Zeitraum="21.05.2014" Wetterlage="sonnig">
  <Stadt>Wien</Stadt>
  <Land>Österreich</Land>
  <PLZ>1230</PLZ>
  <Regenwahrscheinlichkeit>0%</Regenwahrscheinlichkeit>
  <Windgeschwindigkeiten>10km/h</Windgeschwindigkeiten>
  <Bewölkung>5%</Bewölkung>
  <Temperatur Grenze="min" Wert = "16"/>
  <Temperatur Grenze="max" Wert = "24"/>
</Wetter>
<Wetter Zeitraum="22.05.2014" Wetterlage="bewölkt">
  <Stadt>Wien</Stadt>
  <Land>Österreich</Land>
  <PLZ>1230</PLZ>
  <Regenwahrscheinlichkeit>30%</Regenwahrscheinlichkeit>
  <Windgeschwindigkeiten>55km/h</Windgeschwindigkeiten>
  <Bewölkung>75%</Bewölkung>
  <Temperatur Grenze="min" Wert = "10"/>
  <Temperatur Grenze="max" Wert = "19"/>
</Wetter>
</Wetterservice>
```

JSON-Datei (655 Bytes):

```
{
 "Wetter": [
  { "Zeitraum": "21.05.2014",
  "Wetterlage": "sonnig",
  "Stadt": "Wien",
  "Land": "Österreich",
```

```
  "PLZ": "1230",
  "Regenwahrscheinlichkeit": "0%",
  "Windgeschwindigkeit": "10km/h",
  "Bewölkung": "5%",
  "Temperatur": [
   {"Grenze": "min", "Wert": "16"},
   {"Grenze": "max", "Wert": "24"}
  ]},
  { "Zeitraum": "22.05.2014",
  "Wetterlage": "sonnig",
  "Stadt": "Wien",
  "Land": "Österreich",
  "PLZ": "1230",
  "Regenwahrscheinlichkeit": "30%",
  "Windgeschwindigkeit": "55km/h",
  "Bewölkung": "75%",
  "Temperatur": [
   {"Grenze": "min", "Wert": "10"},
   {"Grenze": "max", "Wert": "19"}
   ]}
  ]
}
```


Wie man sehen kann, enthalten beide Dateien denselben Inhalt. Der Inhalt in der JSON-Datei nimmt nur 655 Bytes ein, der in der XML-Datei 763 Bytes. Umso kompakter die Daten sind, umso weniger Daten müssen übertragen werden. Dies wird in ländlicheren Gegenden zum großen Vorteil, da es nicht überall schnelle Internetverbindungen gibt. (siehe Kapitel 4.6)

Die Umsetzung der verschiedenen Technologien in der Android Umgebung wird nun in zwei unterschiedlichen Beispielen dargestellt:

JSON kann man in Java in zwei Richtungen einsetzen: Java Klassen in JSON data konvertieren (Serialization) oder parse JSON data und erstelle eine Java Klasse (Deserialization). In diesem Fall wird die Methode „Deserialization“ angewendet, um Daten aus dem Webservice auszulesen und in JSON Objekte umzuwandeln. Dies kann man mit eingebetteten Funktionen und dem „JSONObject“ durchführen. Falls man eine komplexe JSON Struktur vor sich hat und Auslesemethoden nicht selbst schreiben möchte, kann man sich zum Beispiel den Onlinegenerator „jsonschema2pojo“[40] zu Hilfe nehmen. Dort kann man ein JSON Beispiel angeben und die Homepage liefert den dazu

[40] Jsonschema2pojo (POJO) kann JSON Dokumente in Java-Funktionen umwandeln. POJO steht für „plain old java object“ und bedeutet „ganz normales java Objekt“.

passenden Android Java Code dazu zurück. Dieser sieht im Fall des Beispiels mit dem Wetterobjekt wie folgt aus:

```
JSONObject wetterservice = new JSONObject(data);
JSONArray wetter = wetterservice.getJSONArray("Wetter");
for (int i=0; i < wetter.length(); i++) {
  JSONObject singleO = wetter.getJSONObject(i);
  String zeitraum = singleO.getString(“Zeitraum“);
  String wetterlage = singleO.getString(“Wetterlage“);
  String stadt = singleO.getString(“Stadt“);
  String land = singleO.getString(“Land“);
  String plz = singleO.getString(“PLZ“);
  String regen = singleO.getString(“Regenwahrscheinlichkeit“);
  String wind = singleO.getString(“Windgeschwindigkeit“);
  String bewoelkung = singleO.getString(“Bewölkung“);
  JSONArray temperatur = singleO.getJSONArray("Temperatur");
  for (int j=0; j < temperatur.length(); j++) {
    JSONObject temp = temperatur.getJSONObject(i);
    String grenze = temp.getString(“Grenze“);
    String wert = temp.getString(“Wert“);
  }
}
```

Zu Beginn wird das JSON-Objekt initialisiert und damit das komplette Objekt in einer Zeile in der Android Umgebung zur Verfügung gestellt. Nach dieser Zeile befasst sich der Code nur damit, die Daten auszulesen. In wenigen Codezeilen kann man jede Information aus der JSON Datei auslesen und entweder ausgeben oder zur Weiterverarbeitung benutzen. In diesem Fall hat man den Inhalt der JSON-Datei gekannt und konnte so die richtigen Attribute auslesen. Wenn man noch nicht weiß, welches JSON Objekt man erwarten kann, ist das Auslesen überaus kompliziert. In dem Fall müsste man sich mit der Methode „names()“ alle Namen der Elemente des JSON-Objekts in einem Array ausgeben lassen. Danach muss man die Daten dann in Schleifen auslesen und unterscheiden, ob es sich um ein Objekt, ein Array oder eine einfache Zeichenkette handelt. Die Struktur des Objekts zu kennen erleichtert die Programmierarbeit um ein Vielfaches.

Die gleichen Funktionen sehen mit XML in Android wie folgt aus:

```
Document xmlFile = new SAXBuilder().build( "wetter.xml" );
Element wetterservice = doc.getRootElement();
List<?> wetter = wetterservice.getChild("wetter");
for (int i=0; i < wetter.size(); i++) {
  Element singleE = wetter.get(i);
  String zeitR = singleE.getAttribute(“Zeitraum”).getValue();
  String wetterL = singleE.getAttribute(“Wetterlage”).getValue();
```

```
  String stadt = singleE.getChildText(“Stadt“);
  String land = singleE.getChildText(“Land“);
  String plz = singleE.getChildText(“PLZ“);
  String regen = singleE.getChildText(“Regenwahrscheinlichkeit“);
  String wind = singleE.getChildText(“Windgeschwindigkeit“);
  String bewoelkung = singleE.getChildText(“Bewölkung“);
  List<?> temperatur = singleE.getChild("Temperatur");
  for (int j=0; j < temperatur.size(); j++) {
    Element temp = temperatur.get(i);
    String grenze = temp.getAttribute(“Grenze“).getValue();
    String wert = temp.getAttribute(“Wert“).getValue();
  }
 }
```

Man kann sehen, dass der Unterschied zwischen XML und JSON in der Android Java Programmierung sehr gering ausfällt. Die unterschiedlichen Methoden heißen anders und ein paar der XML-Auslesemethoden benötigen den Zusatz“getValue()“, um den Wert eines Attributes zu erhalten. Auch hier gilt, dass es nur so wenig Code sein kann, wenn der Programmierer die Struktur der XML-Datei kennt. Falls dies nicht bekannt ist, kann man mit Hilfe der Methode „getChildren()“ alle Elemente der XML-Datei in ein List-Array speichern.

Wie man aus den beiden Programmierbeispielen sehen kann, ist die Programmierung mit XML oder JSON von der Anzahl der Codezeilen fast ident und auch die Komplexität in beiden Fällen ist sehr ähnlich. Die Entscheidung, das Webservice mit REST umzusetzen, hat keine Auswirkung auf die Entscheidung zwischen JSON und XML. REST kann beide Technologien übertragen. JSON ist im Gegensatz zu XML etwas einfacher und flexibler einzusetzen. Maschine und Mensch kann die JSON-Datei einfach lesen und der große Vorteil der geringeren Datengröße war hier auschlaggebend, um die Entscheidung zu Gunsten von JSON zu treffen.

4.6. 2G vs. 3G vs. 4G

Heutzutage gibt es aktuell drei unterschiedlich schnelle Telekommunikationsnetze. Die unterschiedliche Geschwindigkeit der Datenübertragung benötigt unterschiedlich viel oder wenig Strom. Umso weniger Daten übertragen werden sollen, umso langsamer kann die Geschwindigkeit für die Datenübertragung sein. In den folgenden Unterkapiteln wird dies genauer erklärt.

Der Datentransfer in den Telekommunikationsnetzen wird momentan mit Hilfe von drei unterschiedlichen Technologiestandards ermöglicht. In Zukunft soll es auch noch einen vierten, noch schnelleren Standard namens LTE-Advanced (Long-Term-Evolution-Advanced), geben, der auf 4G (LTE) aufsetzt.

4.6.1. GSM (Global System for Mobile Communications)

Das GSM-Netz stellt die zweite Generation des Mobilfunks dar. Die erste Generation beschränkte sich auf analoge Netze und die letzten Vertreter dieser Generation wurden in Europa um die Jahrtausendwende abgeschaltet. Daher wird die erste Generation des Mobilfunks hier nicht berücksichtigt. Der GSM-Standard war notwendig, um das erste volldigitale Mobilfunknetz zu ermöglichen. Die Idee dahinter war europaweit Mobilfunkteilnehmer zu vernetzen und zusätzlich eine Schnittstelle zu den ISDN oder analogen Telefonnetzen bereitzustellen. Im Jahr 2006 vermeldete die Mobilfunkbranche, dass das GSM-Netz über 2 Milliarden Anwender aufweist. Bei der damaligen Weltbevölkerung von 6,6 Milliarden Menschen hatte fast jeder 3. Mensch bereits 2006 ein Mobiltelefon mit Zugang zu einem GSM-Netz. (2 Mrd. GSM Nutzer)

Die Datenübertragung im GSM-Standard funktioniert über eine Mischung aus Frequenzmultiplexing (FDM) und Zeitmultiplexing (TDM). Multiplexing ist eine Technologie, bei der mehrere Signale gebündelt werden und gleichzeitig übertragen werden können. Diese Übertragung muss nicht immer über einen Funkturm stattfinden, sondern kann auch über Kabel geschehen. Die FDM-Technologie ermöglicht es gleichzeitig mehrere Signale auf mehrere Träger zu verteilen. Jeder Träger kann mehreren Frequenzen zugeordnet sein und kann daher auch mehrere Kanäle empfangen. Durch die Bündelung kann man schnellere Datenübertragungsgeschwindigkeiten erreichen. Die TDM-Technologie ordnet jedem Empfänger einen gewissen Zeitslot in dem Frequenzbereich zu, in dem gesendet werden kann. Bei der Synchronen TDM können alle Geräte gleichzeitig online sein und zeitgestaffelt Daten empfangen. Die Geräte können durch die Positionen erkannt werden. Das Problem bei dieser Technologie ist, dass bei Standbyezeiten Kanäle nicht genutzt werden und die Übertragungsgeschwindigkeit für andere Benutzer nicht erhöht werden kann. Bei der Asynchronen TDM können diese Zeitslots auch von anderen Geräten genutzt werden, jedoch kann man dann durch den Zeitslot nicht mehr eindeutig

erkennen, welches Gerät sich dahinter befindet, und muss in den Datenpaketen die Informationen mitsenden. (Grundlage Mobilfunk)

Die GSM-Technologie kann auf unterschiedlichen Frequenzbereichen senden und jedes Land oder auch jeder Kontinent kann dies individuell entscheiden.

Der GSM-Standard kann nur sehr geringe Übertragungsraten ermöglichen. Die maximale Übertragungsgeschwindigkeit hat sich auf 14,4 Kilobit pro Sekunde (kBit/s) beschränkt. Durch die Anforderungen der Benutzer an das GSM-Netz, immer mehr Daten immer schneller übertragen zu wollen, war schnell klar, dass man hier eine neue Technologie mit schnelleren Übertragungsraten benötigt. Der Benutzer wollte nicht nur mehr telefonieren oder Kurznachrichten verschicken, sondern auch Daten aus dem WAP[41] oder direkt aus dem Internet herunterladen, beziehungsweise auch hochladen. Daraufhin wurden die Technologiestandards GPRS und EDGE entwickelt. (GSM Erfolgsgeschichte)

4.6.2. UMTS (Universal Mobile Telecommunications System)

UMTS wird auch 3G (dritte Generation des Mobilfunks) genannt. Der Standard UMTS verfolgt zum schnelleren Datentransfer einen anderen technischen Ansatz als GSM: Wideband CDMA (Code Division Multiple Access). Mithilfe dieser Technologie kann das Endgerät mehrere Datenströme gleichzeitig senden und empfangen. Mit dem UMTS-Standard bekommt jedes Mobiltelefon einen unterschiedlichen Code, mit dem es identifiziert werden kann. Alle Mobiltelefone nutzen die gesamte Bandbreite des verfügbaren Netzes. Somit können sehr schnell Datentransferraten erzielt werden, wenn das Netz nicht voll ausgelastet ist und man sich die Bandbreite nicht mit vielen anderen Benutzern teilen muss. Falls sich der Benutzer auf einem Event mit vielen anderen Mobilfunkbenutzern befindet, wie etwa auf einem Konzert oder in einem Stadion, kann

[41] Wireless Application Protocol (WAP) ist eine Sammlung von Protokollen und Techniken, die einem Benutzer Inhalte des WWWs zur Verfügung stellen sollen. Hier sei zu beachten, dass diese für Mobiltelefondisplays optimiert und die Datenübertragung möglichst gering gehalten werden sollte. Die WAP Technologie fand ihre Blütezeit rund um die Jahrtausendwende und ist heutzutage im Großteil der Welt wieder verschwunden. Die Technik wurde früher verwendet, um Webinhalte auf Mobiltelefonen mit schwachen Prozessorleistungen und geringen Arbeitsspeicher darstellen zu können. Die übertragene Datenmenge für Inhalte musste durch das vergleichsweise langsame GSM-Netz möglichst gering gehalten werden. Viele Provider von Inhalten haben daher auf große Datenmenge wie etwa Bilder, Musik oder Videos verzichten müssen. (Informatik Duden) (Wirtschaftsinformatik I)

es daher zu sehr langsamen Verbindungen kommen. Der Vorteil bzw. in diesem Fall der Nachteil, dass jedes Gerät immer online ist, verlangsamt hier das UMTS-Netz. Bei Fußballspielen in einem Stadion kann beobachtet werden, dass sich viele Mobilfunktelefone während der Halbzeiten im Idle-Modus[42] befinden. Kurze Zeit vor und nach der Halbzeitpause ist das UMTS-Netz meistens vollkommen ausgelastet oder auch überlastet, da alle Zuschauer in den Halbzeitpausen ihr Mobiltelefon aktivieren und telefonieren, Kurznachrichten verschicken oder im Internet surfen wollen. Meistens stoßen die UMTS-Funkmaste an ihre Grenzen, so dass es zu einer Überlast bzw. einem Absturz des Netzes kommt. (Grundlage Mobilfunk)

Im Vergleich zum GSM-Standard kann UMTS schon sehr schnelle Verbindungsgeschwindigkeiten anbieten mit bis zu 384 kBit/s. Der UMTS-Standard ist auch, wie das GSM-Netz, ein digitales Netz und momentan wird in vielen europäischen Ländern sowohl ein UMTS- als auch ein GSM-Netz betrieben. Aktuelle Smartphones können beide Netze nutzen. Falls nicht anders vom Benutzer festgelegt, verbindet sich das Smartphone automatisch mit einem UMTS-Netz. Falls die Verbindung zum UMTS-Netz sehr schlecht oder gar nicht vorhanden ist, wechselt das Mobiltelefon automatisch in das langsamere GSM-Netz. In vielen europäischen Ländern ist das UMTS-Netz in Ballungszentren sehr gut ausgebaut, aber in ländlichen Gebieten und entlang von Autobahnen ist es nicht so stark verbreitet. Durch die WCDMA-Technologie sinkt die maximale Übertragungsrate, je weiter das Mobiltelefon von einem Funkmast entfernt ist und je schneller es sich von diesem wegbewegt. (Netztest AT) Falls das UMTS-Netz ausgelastet ist, kann auf modernen Smartphones manuell auf das GSM-Netz gewechselt werden, um besseren Empfang für Anrufe, Kurznachrichten oder Internetinhalte zu bekommen. (UMTES & GSM Vergleich)

[42] Idle-Modus („(to) idle“ deutsch: leerlaufen): Ein Mobiltelefon, das nicht aktiv genutzt wird, sondern im Standbye-Modus ausharrt. Das Telefon kann Anrufe, Kurznachrichten oder auch Push-Nachrichten bekommen, verbraucht aber kaum Bandbreite, da es nichts aktiv tut. (Wirtschaftsinformatik I) Push-Nachrichten („(to) push“ deutsch: anstoßen): Sind Nachrichten, die auf ein Endgerät von einem Server oder von einem anderen Endgerät gesendet („gepusht“) und auf dem Empfangsgerät sofort angezeigt werden, ohne dass der Benutzer aktiv nach neuen Nachrichten suchen muss. Viele Mail-Anbieter bieten einen Push-Service an, mit dem Emails automatisch auf das Endgerät des Benutzers gepusht werden, ohne dass der Benutzer aktiv nach neuen Emails suchen muss. (Wirtschaftsinformatik I)

4.6.3. LTE (Long Term Evolution)

Der LTE-Standard baut auf dem UMTS-Standard auf, kann jedoch wesentlich höhere Datenübertragungsraten im Gegensatz zur 3. Generation des Mobilfunks erreichen. Daher wird LTE auch als die 4. Generation des Mobilfunks bezeichnet. Die Übertragungsgeschwindigkeit kann bei LTE bis zu 300 Megabit pro Sekunde (MBit/s) betragen. Der Standard LTE setzt auf Orthogonales Frequenzmultiplexverfahren (OFDM), das eine Sonderform der FDM-Technologie darstellt. Im Prinzip werden bei diesem Verfahren die Daten auf mehrere Teildatenströme aufgeteilt und beim Empfänger wieder zusammengesetzt. Auch wenn diese Teildatenströme kaum schnellere Übertragungsgeschwindigkeiten erzielen können, bekommt man durch die Masse der Teildatenströme, die genutzt werden können, viel bessere Ergebnisse. Der LTE-Standard ist in Europa noch nicht so weit verbreitet wie der UMTS-Standard und das Aufstellen neuer Funkmaste für die LTE-Technologie ist dementsprechend teuer. Auch die LTE-Frequenzen kosteten der Mobilfunkbranche in Österreich unglaubliche 2 Milliarden Euro. (2 Mrd. EUR LTE) Daher wird man vor allem in Österreich noch einige Zeit auf ein flächendeckendes LTE-Netz warten müssen. Die Technologie LTE-Advanced ist abwärts kompatibel zum LTE-Standard und man wird Geräte für LTEA auch im LTE-Netz nutzen können, jedoch muss ein Smartphone LTE-fähig sein, um diesen schnellen Standard zu nutzen. Nur einige wenige Smartphones können dies heutzutage. Die meisten Smartphones verwenden noch den UMTS-Standard und sind noch nicht LTE-fähig[43].

4.6.4. Telekommunikationstechnologien

Dieses Kapitel beschreibt die unterschiedlichen Netze, die von einem modernen Smartphone genutzt werden können. Jedes Netz hat das Ziel, dem Benutzer den Zugang zum Internet über sein mobiles Endgerät zu ermöglichen. Jedes folgende Unterkapitel beschreibt die maximal mögliche Geschwindigkeit, die bei optimalen Bedingungen erreicht werden kann, sowie deren Funktionalität. Im Kapitel 4.6.5 werden diese Technologien miteinander verglichen.

Heutzutage verbindet sich jedes Smartphone automatisch mit der schnellsten Datenverbindung, die verfügbar ist. Es wird keine Rücksicht darauf genommen, ob es

[43] Stand 2014

notwendig ist, Daten schnellst möglich herunterzuladen oder ob das Telefonnetz ausgelastet oder gar überlastet ist. Bei allen gängigen Smartphones kann sich der Benutzer aussuchen, ob 2G, 3G oder 4G als Netz ausgewählt werden soll. Wenn man keine großen Datenpakete versenden oder herunterladen möchte, kann es durchaus hilfreich sein sich im langsameren 2G-Netz zu befinden. Auch wenn die schnellen Netze komplett ausgelastet sind, weil viele Geräte gleichzeitig eingewählt sind, macht es durchaus Sinn ein langsameres Netz auszuwählen. Nicht alle Netzbetreiber stellen in allen Regionen eigene Sendemasten auf. In diesen Gebieten werden dann Kapazitäten von anderen Netzanbietern gemietet, jedoch nur dann, wenn auch wirklich freie Kapazitäten verfügbar sind. Auch hier macht es durchaus Sinn mit der langsameren Verbindung nur geringe Datenpakete herunterzuladen, anstatt mit der schnellen Verbindung keinen Netzzugang zu bekommen.

Das ist einer der Hauptgründe, warum eine Modellierungsapplikation möglichst kleine Datenpakete versenden und erhalten sollte. Wenn der Benutzer sich durch äußere Umstände in einem langsameren Netz befindet, kann er trotzdem problemlos weiterarbeiten.

4.6.4.1. General Packet Radio Service (GPRS)

Geschwindigkeit: 171,2 kBit/s (gleichzeitige Bündelung von 8 Zeitschlitzen zu je 21,4 kBit/s)

Bei der GPRS-Technologie war das Smartphone zum ersten Mal durchgängig online („allways-on") und die Mobilfunkunternehmen haben nach verbrauchter Datenmenge verrechnet und nicht nach der Zeit, die das Telefon online war. Die Technologie GPRS erlaubt die langsamste Datenübertragungsrate und ist zumindest in Österreich kaum mehr im Mobilfunknetz in Verwendung. Die Technologie basierte auf dem GSM-Netz und konnte durch die Bündelung der Zeitslots vergleichsweise hohe Datenübertragungsgeschwindigkeiten erzielen. (GPRS)

4.6.4.2. Enhanced Data Rates for GSM Evolution (EDGE)

Geschwindigkeit: 236 kBit/s

Die Technologie EDGE basiert auch auf dem GSM-Standard und der GPRS-Datenübertragung. Bei EDGE werden verbesserte Modulationsverfahren verwendet, um mehr Daten über denselben Kanal übertragen zu können. (EDGE)

4.6.4.3. High Speed Packet Access (HSPA)

Geschwindigkeit: 14,4 MBit/s

Die Technologie HSPA basiert auf dem UMTS-Standard. Die Datenübertragung funktioniert auch hier über Kanäle und es können bis zu 3 UMTS-Zeitslots gleichzeitig verwendet werden. Außerdem können bis zu 15 Kanäle gleichzeitig von einem Gerät in Anspruch genommen werden.

4.6.4.4. Long Term Evolution Advanced (LTE-Advanced)

Geschwindigkeit: 1 GBit/s

LTE-Advanced soll sogar über 40 MHz Bandbreite verfügen. Diese enorme Frequenzbreite würde sehr schnelle Datenübertragungsraten von bis zu 1 GBit/s erlauben. LTE-Advanced setzt auf die Carrier Aggregation Technologie, mit der Datenraten pro Benutzer erhöht werden können. Durch eine bessere Ausnutzung der Ressourcen können erhöhte Datenübertragungsraten ermöglicht werden. Diese Technologie kam bereits bei dem Upload für HSPA (High Speed Uplink Packet Access (HSUPA)) zum Einsatz. Zusätzlich sollen bei LTE-Advanced mehr Antennen zum Einsatz kommen, sowohl in den Sendemasten als auch in den Smartphones. Aktuell haben Hersteller von Smartphones aber noch nicht wirklich einen Weg gefunden, wie man die Akkulaufzeiten beibehalten und trotzdem mehrere Antennen mit Strom versorgen kann, um die schnelleren Datenübertragungsraten zu realisieren. Diese Technologie wird Multiple Input & Multiple Output (MIMO) genannt. (LTE Technik)

4.6.5. Vergleich

Der Vergleich der unterschiedlichen Telekommunikationstechnologien wirkt relativ einfach:

LTE, als die schnellste Technologie, ist natürlich die präferierte Lösung für die Applikation.

Man sollte nur immer bedenken, dass je schneller die Datenübertragung ist, umso mehr Strom verbraucht das mobile Gerät. Daher ist es wichtig, dass man nicht zu viele Daten herunterladen muss.

Folgende Tabelle zeigt noch einmal zusammengefasst die Unterschiede der verschiedenen Technologien (LTE-Advanced wurde hier nicht mit in die Tabelle aufgenommen, da es noch nicht im Einsatz ist):

	GSM	UMTS	LTE
Generationen	2	3	4
Genutzte Technologien	Zeitslot	WCDMA	OFDM
Genutzte Frequenzen in MHz	0,2	5	20
Übertragungsgeschwindigkeit in MBit/s	0,236	42,2	300
Datenübertragung Kilobyte (KB) /Sekunde	29,5	5275	37500
Datenübertragung Megabyte (MB) /Sekunde	0,0295	5,275	37,5

Tabelle 7 zeigt einen Vergleich der 3 gängigen Telekommunikationstechnologien

In der Zeile 4 der Tabelle 7 kann man die Datenübertragungsgeschwindigkeiten der drei Technologien erkennen. In der folgender Zeile kann man die Datenmenge sehen, die während einer Sekunde übertragen werden kann. Die Tabelle 8 gibt einen Überblick darüber, wie lange es dauert, um eine bestimmte Datenmenge herunterzuladen:

Dateigröße in MB	GSM	UMTS	LTE
0,5	16,949	0,095	0,013
1	33,898	0,190	0,027
10	338,983	1,896	0,267
25	847,458	4,739	0,667
50	1694,915	9,479	1,333
100	3389,831	18,957	2,667
250	8474,576	47,393	6,667
500	16949,153	94,787	13,333
	Zeit in Sekunden		

Tabelle 8 zeigt, wie lange die Datenübertragung bei maximaler Übertragungsgeschwindigkeit dauert

Diese Werte wurden im folgenden Diagramm grafisch aufbereitet:

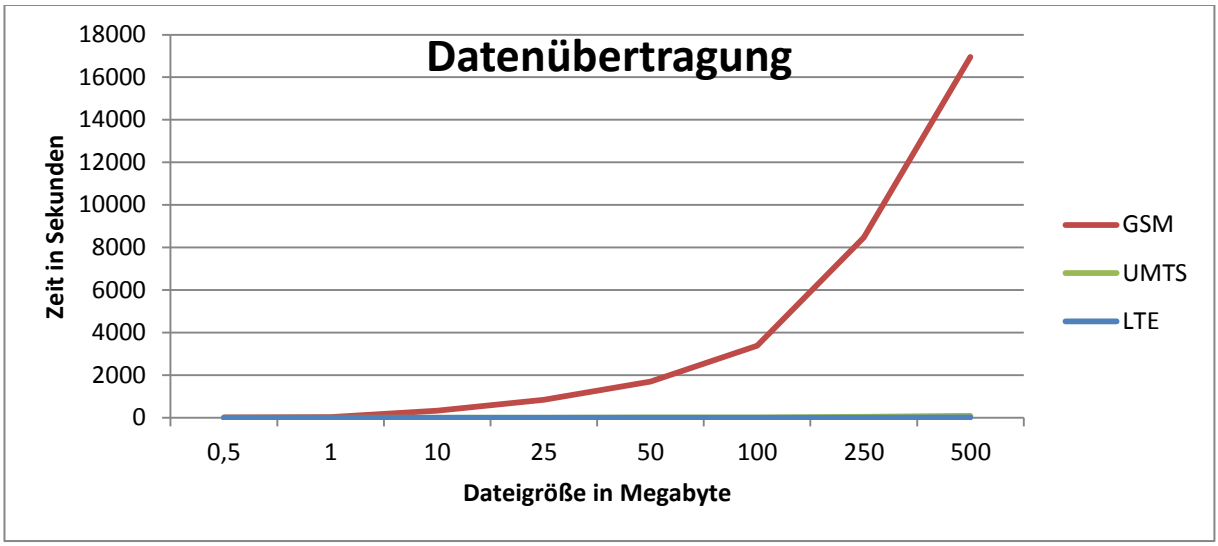


Abbildung 43 veranschaulicht ein Datendiagramm mit 3 Kurven, die die Dauer der Datenübertragung widerspiegeln

Man kann aus dem obigen Diagramm den Unterschied zwischen UMTS (grüne Kurve) und LTE (blaue Kurve) nicht wirklich erkennen, daher wurde im nächsten Diagramm nur noch UMTS und LTE verglichen:

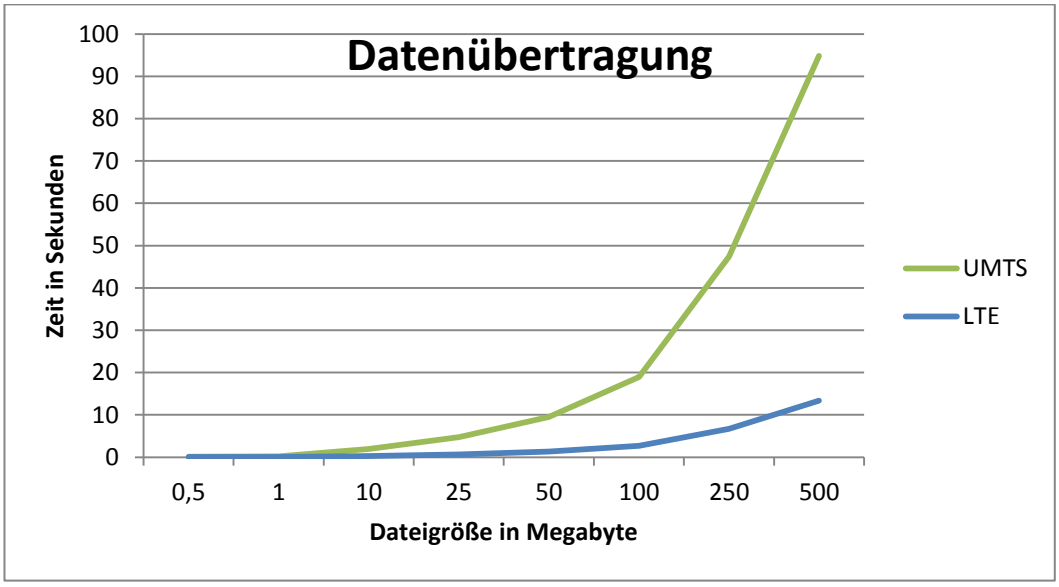


Abbildung 44 veranschaulicht ein Datendiagramm mit 2 Kurven, die die Dauer der Datenübertragung widerspiegeln

Falls LTE nicht verfügbar ist, wäre natürlich UMTS die bevorzugte Lösung, da auch UMTS wesentlich schneller ist als GSM. JSON-Objekte, die Modelle enthalten, werden natürlich keine 500 MB haben, aber Modelle mit bis zu 10 MB kann man durchaus übertragen. Hier beträgt der Unterschied zwischen LTE und UMTS nur 1,6 Sekunden, jedoch ist der Unterschied zwischen UMTS und GSM 337 Sekunden, was schon mehr als 5 ½ Minuten entspricht. Falls man weder LTE- noch UMTS-Netz empfangen kann, ist es also sehr wichtig, dass man möglichst geringe Datenmengen überträgt.

5. Modell-Datenstruktur für Smartphones

Dieses Kapitel befasst sich mit der Struktur und der Speicherung von essentiellen und nicht-essentiellen Daten auf dem Smartphone. Bei jeder adaptierten Modellierungsnotation gibt es essentielle Daten, die zur Anzeige eines Modells notwendig sind. Weiters gibt es nicht-essentielle Daten, die erst durch lazy loading bei Bedarf nachgeladen werden. Durch die Möglichkeit des lazy loading kann die benötigte Datenmenge möglichst gering gehalten werden.

Da man in der heutigen Zeit relativ viel Speicherplatz auf Smartphones zur Verfügung hat, könnte man davon ausgehen, dass der Speicherplatz, den die Applikation benötigt, nicht relevant wäre. Jedoch werden alle Applikationen immer aufwendiger und damit größer. Fotos, die mit dem Smartphone aufgenommen werden, benötigen immer mehr Platz und auch Musik und Videos auf dem Smartphone nehmen zu. In der online Ausgabe der Zeitung „derStandard“ wurde berichtet: „Rund die Hälfte aller Smartphone-Besitzer nutzt ihr Gerät zum Musikhören“ (Zitat aus (derStandard Musik) Absatz 4). Daher sollte man sich bei der Entwicklung einer Applikation Gedanken zum Speicherplatz machen. Die Datenstruktur und die Art, wie Daten gespeichert werden, ist sehr wichtig, da man Applikationen immer wieder beendet und danach dort weitermachen möchte, wo man zuvor aufgehört hat. Dies soll durch das Zwischenspeichern von Daten in der Applikation ermöglicht werden. (Algorithmen & Datenstrukturen) (Skriptum Algorithmen & Datenstruktur)

5.1. Datenstruktur

Wie bereits in der Einleitung erwähnt, gibt es unterschiedliche Anforderungen, die eine Datenstruktur in mobilen Umgebungen zu erfüllen hat.

- Schneller Datenzugriff: Damit ein Benutzer möglichst schnell mobil auf Modelle zugreifen kann, muss ein schneller Datenzugriff gewährleistet werden
- Stabil gegen Verbindungsabbrüche: Um reibungsloses Arbeiten zu ermöglichen, muss eine Applikation auch bei Verbindungsabbrüchen weiterarbeiten können. Daten, die über ein Webservice geladen werden, müssen daher auf dem Smartphone zumindest vorübergehend gespeichert werden.
- Unterscheidung essentielle / nicht-essentielle Daten: Es muss klar definiert sein, welche Daten essentiell für eine Anzeige des Modells sind und welche Daten on-

demand geladen werden können. Diese Anforderung muss auch in der Datenstruktur abgebildet werden. Ein sehr übersichtliches Beispiel ist eine Prozesslandkarte wie in folgender Abbildung:

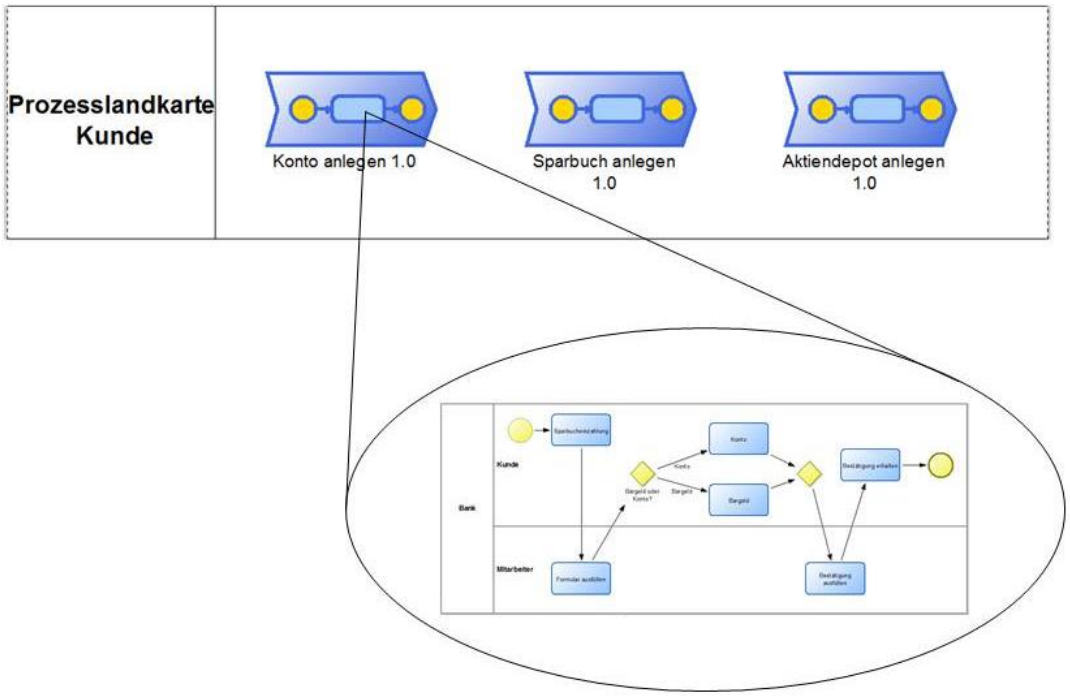


Abbildung 45 zeigt eine Prozesslandkarte und einen dazugehörigen Prozess.

Die Prozesslandkarte „Prozesslandkarte Kunde“, in der Abbildung 45, enthält drei Prozesse, deren Daten essentiell für das Anzeigen des Modells sind. Die drei Prozesse haben jeweilige Geschäftsprozessmodelle hinterlegt, die für die erste Anzeige nicht-essentielle Daten werden erst nach Anklicken des jeweiligen Prozesses durch den Benutzer nachgeladen.

Unter dem Begriff Datenstruktur versteht man die Organisation und Speicherung von Daten. Bei der Datenstrukturierung geht es um die Verwaltung einer Menge ähnlicher Objekte (Daten). Die Adressierung der Daten soll ermöglicht werden und es soll ein schneller und effizienter Zugriff gewährleistet werden. In der heutigen Zeit muss man auf immer größere Mengen an Daten immer schneller zugreifen können, daher müssen Daten strukturiert werden. Unterschiedliche Daten benötigen unterschiedliche Arten von Strukturen, daher benötigen alle Arten von Daten und Datentypen eine andere

Struktur, um den Anforderungen gerecht zu werden. (Skriptum Algorithmen & Datenstruktur)

Ein einfaches Feld ist die einfachste Art der Strukturierung. Ein Feld A[n] besteht aus "n" Variablen (A[1], A[2], ..., A[n]) vom selben Typ. Man kann direkt auf alle Variablen zugreifen, indem man diese direkt adressiert zum Beispiel A[3]. Man kann in diesem Feld auch nach einem Wert suchen und einfach in jedem Feld nachsehen, welchen Wert es enthält, diesen vergleichen und im Ungleichheitsfall das nächste Feld ansehen. Diese Methode kann sehr lange dauern, da im schlechtesten Fall der Wert gar nicht gefunden wird oder erst in der letzten Stelle vorkommt. Daher würde das Programm volle "n" Durchläufe brauchen, um ein Ergebnis zu liefern. Mithilfe eines Suchalgorithmus könnte man diesen Wert viel schneller ermitteln. Wenn man zum Beispiel eine Zahl sucht und das Feld aufsteigend oder absteigend sortiert ist, würde der Algorithmus in der Mitte der Zahlenreihe nachsehen, welcher Wert dort gespeichert ist, und dann entscheiden, ob er im rechten oder im linken Zahlenbereich weitersucht. Diese Methode führt im Durchschnitt viel schneller zu einem Ergebnis, als einfach jedes Feld anzusehen und eins weiterzugehen. Wegen der unterschiedlichen Suchalgorithmen ist es auch wichtig zu entscheiden, welche Datenstruktur am besten geeignet ist. Neben den einfachen Feldern gibt es auch noch ausgereiftere Möglichkeiten Daten zu speichern. Folgen von Elementen, an denen man nur vorne oder hinten ein Element anfügen darf, nennt man „Keller“ bzw. „Schlangen“. Einer Liste, die eine sortierte Menge an Elementen enthält, ermöglicht an jeder Position einen Elementenaustausch durchzuführen. Für Zahlenwerte am besten geeignet sind sogenannte Baum-Strukturen. Diese ermöglichen den schnellst möglichen Zugriff auf eine Zahl in einem sortierten Umfeld. (Algorithmen & Datenstruktur) (Java Insel 9)

Die Datenstruktur auf mobilen Endgeräten ist größtenteils bereits durch das Betriebssystem vorgegeben. Die folgenden automatischen Datenanbindungen sind bereits vorhanden:

- Google's Android: SQLite
- Apple's iOS: SQLite
- Microsoft's Windows Phone: Microsoft (MS) SQL (bzw. LINQ to SQL)

Mit diesen bereits vorhandenen Datenbanken kann man Daten dauerhaft auf dem Smartphone speichern, ohne dass diese bei jedem Start der Applikation von einem Webservice neu geladen werden müssen. Diese bereitgestellte Datenbankanbindung ist überaus schnell, stabil und benötigt keine weiteren Programmbibliotheken oder Adaptionen. (Android DB) (iOS DB) (Windows Phone DB)

Die SQLite Datenbank wird in einer einzigen Datei abgespeichert und stellt eine relationale Datenbank[44] dar, die zur elektronischen Datenverwaltung auf Computern, oder in diesem Fall Smartphones, dient. (Android DB) (iOS DB)

Die MS SQL Datenbank ist auch eine relationale Datenbank. Diese wird von Windows Phone per LINQ to SQL angesprochen, die Anfragen als Objekt in SQL umwandelt und Antworten umgewandelt in ein Objekt zurückgibt. (Windows Phone DB)

5.2. Speicherung der Daten und Anforderungen an die Datenbank

Durch die einfache Anbindung an die SQLite bzw. MS SQL Datenbank ist die Art der Speicherung der Daten quasi vorgegeben. Die Datenbanken bieten die benötigte Funktionalität, damit die Applikation einwandfrei arbeiten kann. Die folgenden Beispiele werden in SQLite gehalten, da diese Lösung von zwei großen Smartphonebetriebssystemen eingesetzt wird.

Anforderungen der Applikation:

- Hinzufügen der Daten
- Entfernen der Daten
- Auslesen der Daten
- Zählen von Tabelleneinträgen

Wie diese Anforderungen in einer Programmiersprache umgesetzt werden, wurde im Anhang in Kapitel 13.2Datenbankanforderungen näher beschrieben.

Sobald eine Applikation auf einem Smartphone installiert wird, werden folgende Tabellen zur Datenspeicherung benötigt:

[44] Einer relationalen Datenbanken liegt die relationale Algebra zu Grunde. Das heißt Operationen auf der Datenbank werden mithilfe der relationalen Algebra durchgeführt.

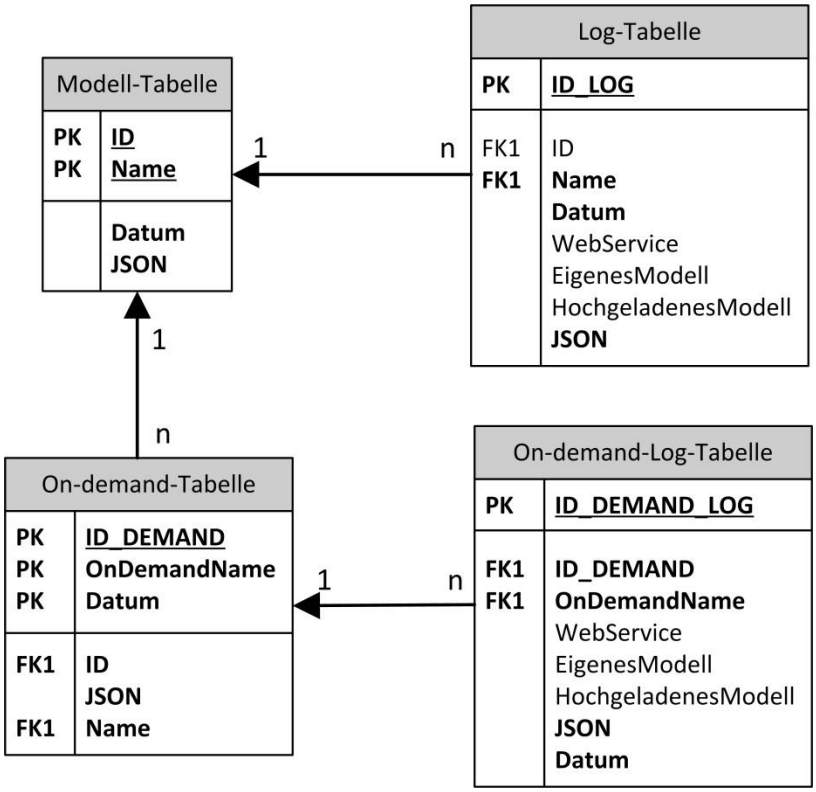


Abbildung 46 zeigt das Entity-Relationship Modell der benötigten Datenbanktabellen.

In Abbildung 46 kann man die Tabelle „Modell-Tabelle“ erkennen, die alle essentiellen Daten zur Darstellung eines Modells auf dem Smartphone speichert. Die Tabelle hat folgende Pflichtfelder:

„ID“: Ist ein automatischer einzigartiger Zähler, der als Primärschlüssel fungiert.

„Name“: Zeigt den Namen des Modells, der als Primärschlüssel fungiert.

„Datum“: Enthält den Zeitstempel, zu dem der Eintrag erfasst wurde.

„JSON“: Beinhaltet die JSON-Datei, die von einem Webservice übergeben wurde.

Die Tabelle „Log-Tabelle“ aus der Abbildung 46 speichert in regelmäßigen Abständen Sicherheitskopien von dem aktuellen Modell, um Datenverlust zu verhindern. Die „Log-Tabelle“ hat folgende Felder:

„ID_LOG“ (Pflichtfeld): Ist ein automatischer einzigartiger Zähler, der als Primärschlüssel fungiert.

„ID“: Ist ein Fremdschlüssel der Tabelle „Modell-Tabelle“.

„Name“ (Pflichtfeld): Ist ein Fremdschlüssel der Tabelle „Modell-Tabelle“ und zeigt den Namen des Modells.

„Datum“ (Pflichtfeld): Enthält den Zeitstempel, zu dem der Eintrag erfasst wurde.

„Webservice“: Soll zeigen, ob ein Eintrag über einen Webservice gekommen ist.

„EigenesModell“: Soll zeigen, ob ein Eintrag auf dem Smartphone erzeugt wurde.

„HochgeladenesModell“: Soll zeigen, ob ein Eintrag auf den Server hochgeladen wurde.

„JSON“ (Pflichtfeld): Beinhaltet die JSON-Datei, die von einem aktuellen Modell als Sicherungskopie angelegt wird.

Die Tabelle „On-demand-Tabelle“ aus der Abbildung 46 speichert „lazy loading“ Anfragen des Benutzers in einer Datenbanktabelle ab. Diese Daten enthalten Zusatzinformationen zu den Modellen. Die „On-demand-Tabelle“ hat folgende Felder:

„ID_DEMAND“ (Pflichtfeld): Ist ein automatischer einzigartiger Zähler, der als Primärschlüssel fungiert.

„OnDemandName“ (Pflichtfeld): fungiert als Primärschlüssel und als Namen des on-demand-Aufrufs.

„Datum“ (Pflichtfeld): fungiert als Primärschlüssel und enthält einen Zeitstempel vom Aufruf.

„ID“: Ist ein Fremdschlüssel der Tabelle „Modell-Tabelle“.

„Name“ (Pflichtfeld): Ist ein Fremdschlüssel der Tabelle „Modell-Tabelle“ und zeigt den Namen des Modells.

„JSON“ (Pflichtfeld): Beinhaltet die JSON-Datei, die von einem aktuellen Modell als Sicherungskopie angelegt wird.

Die Tabelle „On-demand-Log-Tabelle“ aus der Abbildung 46 soll die Daten von der „On-demand-Tablle“ zwischenspeichern, damit bei einem Abbruch der Applikation oder einem Verbindungsabbruch nicht alle Daten, die bereits schon einmal geladen wurden, noch einmal geladen werden. Weiters soll diese Tabelle Änderungen an den Details, die durch den Benutzer durchgeführt wurden, mitspeichern. Die „On-demand-Log-Tabelle“ hat folgende Felder

„ID_DEMAND_LOG“ (Pflichtfeld): Ist ein automatischer einzigartiger Zähler, der als Primärschlüssel fungiert.

„ID_DEMAND“: Ist ein Fremdschlüssel der Tabelle „On-demand-Tabelle“.

„OnDemandName“ (Pflichtfeld): Ist ein Fremdschlüssel der Tabelle „On-demand-Tabelle“ und zeigt den Namen des Aufrufs.

„Webservice“: Soll zeigen, ob ein Eintrag über einen Webservice gekommen ist.

„EigenesModell“: Soll zeigen, ob ein Eintrag auf dem Smartphone erzeugt wurde.

„HochgeladenesModell“: Soll zeigen, ob ein Eintrag auf den Server hochgeladen wurde.

„JSON“ (Pflichtfeld): Beinhaltet die JSON-Datei, die von einem aktuellen Modell als Sicherungskopie angelegt wird.

„Datum“ (Pflichtfeld): Enthält den Zeitstempel zu dem Eintrag, der erfasst wurde.

Die „On-demand-Tabelle“ wird für einen flüssigen Betrieb der Applikation benötigt. Sobald diese Information am Smartphone gespeichert sind, kann man diese zusätzlichen Informationen im Modell immer wieder verstecken oder anzeigen, ohne dass die Applikation jedes Mal die Daten aufs Neue laden muss.

Damit man nicht zu viel Platz auf einem Smartphone mit der Applikation belegt, kann man in den Benutzereinstellungen festlegen, wie viele der letzten geladenen Modelle auf dem Gerät gespeichert bleiben sollen. Da es bei einem Smartphone immer wieder zu einem Netzausfall kommen kann oder der Akku des Geräts leer wird, speichert die Applikation in regelmäßigen Abständen Sicherheitskopien in der "Log-Tabelle" ab. In den Benutzereinstellungen kann eingestellt werden, nach wie vielen Minuten eine Sicherung angelegt werden soll. Weiters kann festgelegt werden, wie viele Versionen

eines Modells auf dem Smartphone verbleiben sollen. Um diese Funktionalität zu gewährleisten, ist es eine Zählmethode durchaus von Vorteil.

Wie solche Tabellen mit Programmcode erstellt werden, sieht man im Anhang in Kapitel 13.3 Erstellen einer Tabelle mit Android.

Zum Speichern von Daten in der Datenbank sollte man in Android eigene Methoden definieren. Man kann mithilfe der bereits vorhandenen SQLite-Methode für Android „insert“ einen Datensatz in die Datenbank speichern. Um die Methode „insert“ sinnvoll mit Daten zu befüllen und nicht für jeden Eintrag neue Codezeilen schreiben zu müssen, wäre es gut dies in eine Methode auszulagern. Am effektivsten ist es, wenn man dieser selbst implementierten Methode alle benötigten Daten in einem selbst definierten Objekt mitgibt, um damit dann die „insert“ Methode zu befüllen.

Um Daten aus der Datenbank zu löschen bzw. Daten auszulesen, geht man genauso vor wie beim Hinzufügen. Zum Auslesen gibt es die vordefinierte SQLite-Methode für Android „query“ und zum Löschen gibt es die „delete“ Methode.

6. Technologie-, Applikations- und Protokollbewertung

In der Technologiebewertung werden Kriterien und Anforderungen definiert, die ein Benutzer an eine Applikation stellt. Hier geht es nicht nur um modellierungsspezifische Aspekte sondern auch um allgemeine Anforderungen, die eine Applikation erfüllen muss. Im Unterkapitel Protokollbewertung werden die unterschiedlichen Protokolle zur Übermittlung der Daten, die in dieser Arbeit bereits vorgestellt wurden, bewertet.

6.1. Technologie- und Applikationsbewertung

Um eine Applikation bewerten zu können, müssen einige Anforderungen definiert und erfüllt werden. Ein großer Teil dieser Kriterien wurde bereits in der Arbeit erwähnt:

- Werden aktuelle Programmierpattern verwendet, um die Performance der Applikation zu optimieren? (siehe Kapitel 3.2.1.4 und Kapitel 5.1)
- Was passiert bei Netzverlust mit der Applikation? (siehe Kapitel 4.2.1 und Kapitel 5.2)
- Kann nach einem Netzverlust weitergearbeitet werden? (siehe Kapitel 4.2.1 und Kapitel 5.2)
- Wie sieht die Kommunikation der Applikation mit einem Server aus? (siehe Kapitel 4.3)
- Werden die Daten am Gerät gecached? (siehe Kapitel 4.4.3 und Kapitel 5.2)
- Wie groß sind die Datenpakete beim Datentransfer? (siehe Kapitel 4.5.3 und Kapitel 4.6.4)
- Wie sieht der Stromverbrauch der Applikation aus? (siehe Kapitel 4.6)
- Wie schnell geht der Datentransfer von statten und wie sieht dieser in der Anlehnung an den Punkt "Stromverbrauch" aus? (siehe Kapitel 4.6)
- Hat der Benutzer die Kontrolle über die Größe des Caches? (siehe Kapitel 5.2)
- Wie groß kann ein Datenpaket maximal sein und kann es nach einem Netzverlust weitergeladen werden oder beginnt der Download wieder bei null? (siehe Kapitel 5.2)
- Ist die Datensicherheit gewährleistet? (siehe Kapitel 7.1)
- Werden sensible Daten ausreichend geschützt? (siehe Kapitel 7.1)
- Wie sieht der CPU / Arbeitsspeicher / lokal benötigte Speicher der Applikation aus?

- Wie sieht die Usability der Applikation aus in Hinsicht auf Ausführbarkeit auf eine Tablet oder einem Desktop-Computer?

Zu diesen relativ spezifischen Anforderungen kommen noch einige allgemeine Anforderungen, die vor allem auf die Benutzerusability abzielen. (App Usability) Hierbei handelt es sich teilweise um Erwartungen, die ein User an eine Smartphone-Applikation hat und teilweise allgemeine Anforderungen die eine Applikation erfüllen muss:

1. Auf bereits Bekanntes und Erlerntes setzen
   Wissen, das sich der Benutzer der App bereits angeeignet hat, sollte ausgenutzt werden. Standardelemente, die ein Benutzer von Smartphones oder einem PC kennt, sollten zum Einsatz kommen. Ein Beispiel hierfür wäre ein "Zurück-Button", mit dem der Benutzer einen Schritt zurückgehen kann. Dieser Button hat fast immer als Symbol einen Pfeil, der nach links zeigt.
2. (Den geringen) Platz für das Wesentliche nutzen
   Dieser Punkt zielt genau auf das Thema dieser Arbeit ab. Der Benutzer sollte nur essentiellen Daten angezeigt bekommen, die man auf einem kleinen Display gut aufbereiten muss.
3. Gute Sichtbarkeit bedeutet nicht gleich gute Bedienbarkeit
   Auf einem Touchscreen sollten nicht zu viele Links und Buttons direkt nebeneinander liegen, da sich der Benutzer ansonsten sehr leicht verklicken kann.
4. Call-To-Action Elemente beim Wort nehmen
   Inhalte, die zur Interaktion einladen, sollen deutlich gekennzeichnet werden. Falls ein Benutzer aus einer App einen Anruf tätigen kann, soll dies auch deutlich erkennbar sein. Falls man diese Möglichkeit nicht anbieten möchte, sollte man auch keine eindeutigen Symbole nutzen, die zur Interaktion einladen. Wenn man eine Telefonnummer mit einem Anrufhörer-Symbol darstellt, sollte es dem Benutzer möglich sein, beim Anklicken einen Anruf zu starten. Es ist frustrierend für einen Benutzer, wenn er denkt eine Funktion ist vorhanden, aber es passiert nichts, wenn er diese ausführen will.
5. Auf hohes Kontrastverhältnis achten
   Schriften sollten sich eindeutig vom Hintergrund abheben und auf

„Kleingedrucktes“ sollte bei einem kleinen Display sowieso verzichtet werden, weil es für den Benutzer nur sehr schwer erkennbar ist.

6. Dem Nutzer klares Feedback geben
   Wenn der Benutzer etwas anklickt, sollte dies markiert oder herausgehoben werden, damit der Benutzer eindeutig erkennen kann, was gerade in der Applikation passiert.
7. Den Nutzer unterstützen
   Der Benutzer könnte mit einer automatischen Vervollständigung unterstützt werden.
8. Gerätespezifische Funktionalitäten sinnvoll nutzen
   Dieser Punkt beinhaltet auch bereits Erwähntes in den Punkten 1 und 3. Weiters sollte sich das Bild am Display in Hochformat und Querformat drehen, wenn der Benutzer das Gerät um 90° dreht. Im Kapitel 7 wird auf diesen Punkt eingegangen.

(Diese Punkte sind Zitate von (App Usability) Punkt 1-8) In Deutschland hat der TÜV Rheinland eine Methode entwickelt (TÜV Anforderungskatalog), wie Applikationen ein Prüfzeichen bekommen können und damit als vertrauenswürdig und sicher eingestuft werden. Der Anforderungskatalog teilt den Prüfprozess in drei Ebenen ein:

1. Ebene 1: Eigenerklärung und Funktionstest
2. Ebene 2: Kontrolle der Berechtigungen
3. Ebene 3: Bewertung des Datenverkehrs

(Diese 3 Punkte sind Zitate von (TÜV Anforderungskatalog) Seite 4 - 5) Weiters werden in der Quelle (TÜV Anforderungskatalog) folgende Anforderungen definiert:

- Konformität zwischen Eigenerklärung und Test
- Bewertung der Eigenerklärung zu Rahmenbedingungen
- Bewertung der Eigenerklärung zu rechtlichen Vereinbarungen
- Bewertung der Eigenerklärung zum Datenzugriff und zur Datenverwendung
- Bewertung der Eigenerklärung zu Sonderverhalten
- Kriterien zur Bewertung des Datenverkehrs
- Kriterien zur Bewertung der angeforderten Berechtigungen und ihrer Nutzung
- Verschlüsselungsverfahren für die Datenübertragung

- Verschlüsselungsverfahren für das Ablegen sensitiver Daten auf dem Gerät
- Hashverfahren für das Ablegen sensitiver Daten

(Diese Punkte sind Zitate von (TÜV Anforderungskatalog) Seite 5 - 8) In diesem Anforderungskatalog wird ein Schwerpunkt auf die Eigenerklärung der Applikation gelegt. Die Eigenerklärung muss vom Entwickler verfasst werden und beinhaltet die Tätigkeiten und Funktionen, die die Applikation bereitstellt. Es ist auch wichtig, wie der Datenverkehr von statten geht und ob dieser verschlüsselt oder nicht-verschlüsselt abgewickelt wird. Falls Daten verschlüsselt verschickt werden müssen, aber diese nicht-verschlüsselt versendet werden, darf kein Prüfzeichen vergeben werden. Es wird weiters darauf eingegangen, wie die Daten verschlüsselt werden und welche Berechtigungen die Applikation auf dem Smartphone vom Benutzer benötigt. (TÜV Anforderungskatalog)

Bei der Frage, für welches Betriebssystem man eine Applikation entwickeln sollte, kann man die Möglichkeiten nach Kapitel 2.6 grob auf zwei Betriebssysteme: Android und iOS einschränken. Um eine iOS Applikation zu entwickeln, muss man sich für einen Apple Developer Account registrieren. Dieser kostet im Jahr 99$ (bei einem Wechselkurs von 1 Euro = 1,27377 US-Dollar entspricht das etwa 77,72€)[45]. (Apple Developer) Weiters kann die Entwicklung nur auf einem Mac OSX Betriebssystem mit dem eigenen Entwicklertool XCode durchgeführt und der Code mit diesem kompiliert werden. Im Gegensatz dazu kann man mit jedem beliebigen Betriebssystem eine Android Applikation entwickeln. Man benötigt hierfür nur die kostenlose Entwicklungsumgebung eclipse (eclipse Framework). Diese ist komplett plattformunabhängig und kann auf allen gängigen Betriebssystemen zum Einsatz kommen.

Im idealen Fall sollte man eine Applikation für alle Plattformen entwickeln, damit man allen Benutzer die Möglichkeit geben kann eine solche Applikation zu nutzen. Falls man jedoch nur beschränkte finanzielle Ressourcen zur Verfügung hat oder erste Testläufe mit eine Applikation durchführen möchte, sollte man zuerst eher die Entwicklungsumgebung Android ins Auge fassen. (siehe hierzu auch Kapitel 2.5 und 2.6) (Gartner Revenue Apps) Die kostenlose und plattformunabhängige Entwicklung macht Android überaus attraktiv. Der positive Aufwärtstrend und die steil nach oben zeigende

[45] Stand 30.10.2014

Zukunftsprognose von Gartner bescheinigen Android eine mehr als rosige Zukunft. (Guardian Revenue Apps) (Smartphone Sales 2014)

6.2. Protokollbewertung

Die Protokollbewertung konzentriert sich auf die unterschiedlichen Möglichkeiten Protokolle im Zusammenhang mit einer Applikation zu bewerten. Protokolle kommen in folgenden Szenarien während des Betriebs einer Applikation zum Einsatz:

- Wie kommunizieren die Applikationen mit einem Server?

Die Kommunikation zwischen einem Server und einer Applikation kann, wie in Kapitel 4 vorgestellt wurde, auf verschiedene Arten passieren. Die Datenübertragung kann via REST oder SOAP durchgeführt werden. Diese beiden "Protokolle" können unterschiedliche Datenformate enthalten, die die Applikation mit Daten vom Server beladen. Wie bereits in Kapitel 4.4.3 festgestellt wurde, eignet sich die Übertragung via REST momentan besser, um Daten an eine Applikation zu senden. (vergleiche Kapitel 4.4.2 und 4.4.3)

Bei der Protokollbewertung kann auch die Bewertung des Datenformats ein wichtiger Aspekt sein. Die unterschiedlichen Formate wurden im Kapitel 4.5 genauer beschrieben. Aktuell sollte man sich hier für JSON entscheiden.

- Was passiert bei „Datenverlust“?

Eine der wichtigsten Fragen, nicht nur beim Arbeiten auf mobilen Endgeräten, ist, was passiert beim „Datenverlust“. Das Smartphone kann verloren, gestohlen oder durch einen Hardwaredefekt unbrauchbar werden. Wie kann man dann noch auf die Daten, die sich auf dem Gerät befinden oder befunden haben, herankommen? Entweder der Benutzer macht regelmäßig ein Backup auf einem lokalen Computer oder die Applikation bietet ein Online-Backup Szenario an. Daten des Benutzers sollten in regelmäßigen Abständen, am besten vom Benutzer selbst einstellbar, auf einem Server abgelegt werden können.

Weiters stellt sich die Frage, was passiert, wenn die Applikation am Smartphone abrupt beendet wird? Dies kann passieren, wenn die Batterie des Smartphones leer ist oder die Applikation durch einen Software- oder Hardwarefehler beendet wird. Im Idealfall sollte

die Applikation noch rechtzeitig alle aktuellen, vom Benutzer nicht gespeicherten, Daten zumindest auf dem Gerät lokal speichern (siehe Kapitel 5.2). Bei einem Hardwaredefekt ist dies zwar nicht möglich, aber bei den anderen Fällen, in denen eine Applikation abrupt beendet wird, sehr wohl.

- Was passiert beim Verlust der Verbindung des Netzes?

Bei sich in Bewegung befindenden mobilen Endgeräten kann es immer wieder auch zum Netzverlust kommen. Im Idealfall stört dieser Zwischenfall die Applikation nicht. Der einzige Effekt ist, dass der Benutzer keine neuen Daten aus dem Internet laden kann. Falls die Applikation, wie empfohlen, bereits geladene Daten gecached hat, kann der Benutzer mit diesen Daten weiterarbeiten und sobald sich das Smartphone wieder im Telefonnetz befindet, können neue Daten heruntergeladen werden.

7. Ausblick

Es gibt noch viele Möglichkeiten, die eine Modellierungsapplikation dem Benutzer anbieten kann, um ihm das Arbeitsleben zu erleichtern. Es gibt die Möglichkeit direkt von Smartphones aus zu drucken. Hierzu gibt es einige Applikationen wie zum Beispiel von den Druckerhersteller Hewlett Packard (HP), Samsung oder Brother oder auch allgemeine Applikationen, die sich nicht auf einen Hersteller konzentrieren wie etwa „Printer Pro“ oder „ePrint“. Man kann diese Apps miteinander interagieren lassen, um etwas zu drucken. Diese Applikationen ermöglichen einem Benutzer auch fremde WLAN Drucker zu adressieren, solange diese öffentlich zugänglich sind und das Smartphone sie orten kann. Dadurch kann sich der Benutzer seine Modelle sofort ausdrucken.

In der heutigen Zeit wird in allen Bereichen der Informatik Kollaboration, im Sinne der Zusammenarbeit, ganz groß geschrieben. Überall arbeiten viele Leute an denselben Projekten und man arbeitet miteinander an unterschiedlichen Themen. Die Applikation sollte einem Benutzer ermöglichen mit dem Ersteller des Modells möglichst schnell und einfach in Kontakt zu treten. Man kann mit Hilfe der Applikation ermöglichen auf das Telefonbuch des Benutzers zuzugreifen und einen Ersteller direkt anzurufen, um mit ihm über das Modell zu sprechen. Falls der Ersteller nicht im lokalen Telefonbuch gefunden werden kann, sollte die Applikation in der Lage sein on-demand die Telefonnummer des Erstellers aus der Modelldatenbank oder einem Onlinetelefonbuch auszulesen. Der Benutzer kann somit gleich mit dem Ersteller in Kontakt treten. Diese Funktionalität sollte sich nicht nur auf das Telefonieren beschränken sondern auch SMS, E-Mails und Messanger abdecken, damit man auch schriftlich mit dem Ersteller in Kontakt treten kann.

Eine weitere Idee, die dem Benutzer das Arbeiten erleichtern kann, ist das automatische Publishing. Die Applikation soll Modelle bearbeiten und diese gleich wieder per Webservice auf eine globale Datenbank synchronisieren können. Damit kann man andere Benutzer gleich an Änderungen und neuen Ideen teilhaben lassen. Weiters kann man auch überlegen, dass Modelle per Email oder Messenger verschickt werden können, um Meinungen von anderen Benutzern direkt abzufragen. Hier müsste man auch eine Lösung anbieten, wenn der andere Benutzer die Modellierungsapplikation gar nicht installiert hat. Die Modelle sollten trotzdem als Bild für ihn ersichtlich sein. Für Benutzer

mit der Modellierungsapplikation sollte es möglich sein, die Modelle in die Applikation zu laden und dort anzusehen, zu bearbeiten und wieder zurückzuschicken.

7.1. Security Issues

In der Applikation sind bisher noch kaum Security Aspekte berücksichtigt worden. Heutzutage sind einige Systeme von Hackangriffen betroffen und man muss versuchen seine Daten vor diesen Angriffen zu schützen. Einer der wichtigsten Aspekte hierbei ist die Authentifizierung. Um ein Webservice in Anspruch zu nehmen und Daten laden zu können, muss sich ein Benutzer erst authentifizieren und zeigen, dass er berechtigt ist, diese Daten zu sehen. Daten in einem Modell sind meistens sensible Daten und sollten von dritten nicht gesehen oder verändert werden.

Ein weiterer Punkt ist die Verschlüsselung der Daten. Sobald Daten verschlüsselt werden und nur Empfänger und Sender den Schlüssel kennen, ist es für dritte sehr schwierig diese Daten zu entschlüsseln. Dies kann sensible Daten auch bei Datenverlust vor unbefugtem Betrachten oder Verändern schützen.

Bei einer Modellierungsapplikation muss man sich auch die Frage stellen, welche Personen sollen Zugriff auf die Applikation haben. Soll jeder User mit einem Smartphone die Möglichkeit haben, die Applikation auf seinem Gerät installieren zu können oder nicht. Alle mobilen Betriebssysteme haben bei Firmengeräten die Möglichkeit firmeneigene Applikationen, die nicht für alle User öffentlich in einem Applikations-Store zugänglich sind, zu installieren. Damit kann man den Kreis der Benutzer, die die Applikation nutzen, sehr stark einschränken.

8. Zusammenfassung

Diese Arbeit befasst sich mit dem Thema der Modellierung auf mobilen Devices. Modelle werden in der Informatik in vielen unterschiedlichen Bereichen, wie etwa der Softwareentwicklung oder beim Darstellen von Geschäftsprozessen, eingesetzt. In den einleitenden Kapiteln wurde bereits ausführlich erklärt, dass es für den Menschen sehr schwierig ist, auf einem kleinen Device viele Elemente eines Models zu erkennen und den Überblick über das ganze Modell zu bekommen. Da der Mensch auf einen Blick kaum mehr als fünf Objekte erkennen und sich merken kann, sollte man bei der Modellierung auf kleinen Devices wichtige Informationen herausheben und unwichtige Informationen für den ersten Blick verstecken. Der Benutzer soll sich auf die wichtigen Teile des Elements konzentrieren können und erst nachdem ein Überblick über das Modell gegeben wurde, soll der Benutzer mit Detailinformationen konfrontiert werden. Neben der Reduktion von Informationen gibt es noch die Möglichkeit Elemente und Beziehungen farblich zu markieren, die Ansicht auf das Wesentliche zu zentrieren oder auch Modelle textuell aufzubereiten. Unterschiedliche Modellierungssprachen müssen unterschiedlich adaptiert werden. In dieser Arbeit wurden dafür verschiedene Modellierungssprachen analysiert und Adaptionsmöglichkeiten vorgestellt. In Kapitel 3.2.1 wurde das Klassendiagramm beschrieben und der Autor ist zu dem Schluss gekommen, dass die beste Möglichkeit der Adaption die Informationsreduktion oder eine zentrierte Sicht ist. Mit der Informationsreduktion werden die wesentlichen Elemente des Klassendiagramms, die Klassen und deren Beziehungen dargestellt und die weniger wichtigen Informationen, wie etwa Attribute und Methoden, ausgeblendet. Der Benutzer kann durch explizites Anklicken von Klassen diese versteckten Informationen nachladen und anzeigen. Die zentrierte Sicht kann diese Informationsreduktion noch unterstützen, indem der Benutzer eine Klasse auswählt und alle Klassen, die nicht direkt oder indirekt mit dieser Klasse in Beziehung stehen, werden ausgeblendet. Im Kapitel 3.2.2 wurde die Modellierungssprache Geschäftsprozessmodellierung vorgestellt und es wurde die Komplexität dieser Modellierungssprache hervorgehoben. Da die Geschäftsprozessmodellierung mehr als 25 unterschiedliche Elemente und Beziehungen zur Modellierung anbietet und diese Modelle sehr groß werden können, ist der Verfasser zu dem Schluss gekommen, dass die textuelle Darstellung die beste Adaptionsmöglichkeit ist. Im Kapitel 3.2.3 wurde die Use-Case-Modellierung beschrieben und auch hier wurde die zentrierte Ansicht empfohlen.

Durch die Einfachheit der Modellierungssprache gibt es zusätzlich die Möglichkeit das komplette Modell am Smartphone anzuzeigen. Eventuell kann man hier auch durch den Einsatz von Farben dem Benutzer das Modell optisch besser aufbereiten.

Modelle werden über Webservices auf das Smartphone des Benutzers übermittelt. Beste Ergebnisse können mit der REST-Technologie erzielt werden, da man mit diesem Standard schneller Daten übertragen kann. Die Daten werden am besten im Datenstandard JSON verpackt, da man damit kleinere Pakete versenden und der Datenverkehr so geringer gehalten werden kann. Kleinere Pakete können schneller versendet werden und das Smartphone benötigt weniger Batterieleistung, um die Daten zu erhalten. Durch die geringen Datenpakete ist man auch nicht zwingend auf die schnellsten Telefonnetze angewiesen, die die schnellsten Übertragungsraten zur Verfügung stellen. Auch die langsameren Datenübertragungsprotokolle können diese Datenpakete in ansprechender Zeit senden und empfangen. Dies wirkt sich positiv, wenn auch nur geringfügig, auf den Batterieverbrauch aus.

Um die Daten optimal auf ein Smartphone überzuleiten, wurde im Kapitel 5 eine Datenstruktur erstellt. Diese Datenstruktur deckt alle Anforderungen ab, die an die Applikation gestellt werden. Es werden die Eventualitäten des Verbindungsabbruchs mit der Logging-Tabelle abgedeckt. Diese Tabelle ist auch für Abstürze der Applikation zuständig und für den Fall, dass der Benutzer vergisst zu speichern. Um Modelle nicht immer neu laden zu müssen, werden Versionen abgespeichert und falls keine neue Version verfügbar ist, muss das Modell nicht neu geladen werden. Dies reduziert den notwendigen Datentransfer zwischen einer Modelldatenbank und dem Smartphone. Die Datenstruktur hilft weiters die adaptierten Modellierungssprachen mit Textbausteinen, zentrierten Ansichten und Informationsreduktion zu unterstützen.

Im letzten Abschnitt dieser Arbeit wurden einige allgemeine Anforderungen, die an eine Applikation gestellt werden, zusammengetragen. Diese Anforderungen decken sowohl den technischen Hintergrund der Applikation ab, als auch User Interface technische Aspekte.

9. Literaturquellenverzeichnis

(Algorithmen & Datenstruktur) N. Blum, „Algorithmen und Datenstrukturen" 1. Aufl Oldenbourg 2004
(Allgemeine Modelltheorie) H. Stachowiak, „Allgemeine Modelltheorie" 1. Aufl Springer 1973
(Android 2) A. Becker, M. Pant, „Android 2: Grundlagen und Programmierung" 2. Aufl dpunkt.verlag 2010
(Computernetzwerke) A. S. Tanenbaum, „Computernetzwerke" 4. Aufl. Addison-Wesley Verlag 2003
(Datenvisualisierung & DM) D. A. Keim, „Datenvisualisierung und Data Mining" Universität Konstanz und AT&T Shannon Research Labs 2003
(Einstieg in XML) H. Vonhoegen, „Einstieg in XML" 4. Aufl. Galileo Computing, 2007
(Eye movements and scene perception) D. E. Irwin, G. J. Zelinsky, „Eye movements and scene perception: Memory for things observed", Perception & Psychophysics 2002,64(6), 882-895, 2002
(Gabler Lexikon Medienwirtschaft) I. Sjurts, „Gabler Lexikon Medienwirtschaft", 2. Aufl. Gabler, 2010
(Hang und Hohensohn 2003) J. Hang und H. Hohensohn, „Eine Einführung zum Open Source Konzept aus Sicht der wirtschaftlichen und rechtlichen Aspekte". 21-Juli-2003
(Informatik Duden) V. Claus, A. Schwill, „Duden. Informatik. Ein Fachlexikon für Studium und Praxis." Bibliographisches Institut, Mannheim 2003
(Java Insel 5) C. Ullenboom, „Java ist auch eine Insel" 5. Aufl. Galileo Computing, 2006
(Java Insel 7) C. Ullenboom, „Java ist mehr als eine Insel" 7. Aufl. Galileo Computing, 2011
(Java Insel 9) C. Ullenboom, „Java ist auch eine Insel" 9. Aufl. Galileo Computing 2010
(Java Web Services) O. Heuser, A. Holubek, „Java Web Services in der Praxis" 1. Aufl dpunkt.verlag 2010
(Komplexität Domain-Models) C. Atkinson, T. Kühne, „Reducing accidental complexity in domain models" Springer Verlag 2007
(Metamodeling Foundation) C. Atkinson, T. Kühne, „Model-Driven Development: A Metamodeling Foundation" IEEE Computer Society 2003
(Metamodelling Platforms) D. Karagiannis, H. Kühn, „Metamodelling Platforms" Springer-Verlag 2002
(OMG BPMN) OMG, „Business Process Model and Notation (BPMN) Version 2.0" OMG 2011
(OMG Metamodell) OMG, „OMG Meta Object Facility (MOF) Core Specification" Version 2.4.1 Object Management Group 2013
(Open Models Feasibility Study) D. Karagiannis, W. Grossmann, und P. Höfferer, „Open Models Feasibility Study". 01-Sep-2008
(Protocol Engineering) H. König, „Protocol Engineering: Prinzip, Beschreibung und Entwicklung von Kommunikationsprotokollen" 1. Aufl Vieweg+Teubner 2003
(REST) R. T. Fielding, „Architectual styles and the design of network-based software architectures" PhD Dissertation 2000
(Semantik Web) P. Hitzler, M. Krötzsch, S. Rudolph, Y. Sure, „Semantik Web", 1. Aufl. Springer examen.press, Juli 2007
(Semantic Web Visualizing) V. Geroimenko, C. Chen, „Visualizing the Semantic Web: XML-based Internet and Information Visualization" 1. Aufl Springer 2003
(Skriptum Algorithmen & Datenstruktur) E. Schikuta, „Algorithmen und Datenstrukturen 1" Universität Wien Skriptum SS 2006

(Skriptum Modellierung) W. Klas, D. Karagiannis, „Modellierung" Universität Wien Skriptum SS 2008
(Skriptum Unternehmensmodellierung) D. Karagiannis, „Unternehmensmodellierung" Universität Wien Skriptum SS 2006
(Sofies Welt) J. Gaarder, „Sofies Welt" 10. Aufl. Deutscher Taschenbuch Verlag 2002
(Steve Jobs Biografie) W. Isaacson, „Steve Jobs: Die autorisierte Biografie des Apple-Gründers" C. Bertelsmann Verlag 2011
(TÜV Anforderungskatalog) TÜV Rheinland Cert GmbH, „Anforderungskatalog zur Bewertung und Zertifizierung mobiler Apps: Check your App / Datenschutz", Version 1.5, 2013
(UML@Work) M. Hitz, G. Kappel, E. Kapsammer, W. Retschitzegger, „UML@Work" 3. Aufl. dpunkt.verlag, 2005
(URI Generic Syntax) T. Berners-Lee, R. Fielding, L. Masinter, „Uniform resource identifier (uri): Generic syntax" The Internet Society 2005
(Ven et al. 2008) K. Ven, J. Verelst, und H. Mannaert, „Should You Adopt Open Source Software?", IEEE Software, Bd. 25, S. 54-59, Mai 2008
(Visualisieren & Präsentieren) J.W. Seifert, „Visualisieren, Präsentieren, Moderieren" 26. Aufl. Offenbach 2009
(Webservices mit REST) L. Richardson, S. Ruby, „Web Services mit REST" 1. Aufl O'reilly 2007
(Wirschaftsinformatik I) H.R. Hansen, G. Neumann, „Wirtschaftsinformatik I", 8. Aufl. Lucius und Lucius (UTB 802). Stuttgart, 2001
(XML WS) F. Coyle, „XML, Web Services, and Data Revolution 1st Edition", 1. Aufl. Addison-Wesley Professional 2002

10. Internetquellenverzeichnis

(2 Mrd. EUR LTE) derStandard. „LTE-Frequenzauktion bringt über zwei Milliarden Euro". [Online]. Verfügbar: http://derstandard.at/1381369315013/Frequenzauktion-bringt-Republik-Oesterreich-zwei-Milliarden-Euro [Besucht: 01-Aug-2015]
(2 Mrd. GSM Nutzer) Computerwoche. „GSM Association meldet zwei Milliarden Nutzer weltweit". [Online]. Verfügbar: http://www.computerwoche.de/a/gsm-association-meldet-zwei-milliarden-nutzer-weltweit,577443 [Besucht: 01-Aug-2015]
(Abstract Java Code) Oracle. „Abstract Methods and Classes". [Online]. Verfügbar: http://docs.oracle.com/javase/tutorial/java/IandI/abstract.html. [Besucht: 01-Aug-2015]
(Amazon Web Service 1) Amazon. „Amazon Elastic Block Store (EBS)". [Online]. Verfügbar: http://aws.amazon.com/de/ebs/. [Besucht: 01-Aug-2015]
(Amazon Web Service 2) Amazon. „Amazon Elastic Compute Cloud (Amazon EC2)". [Online]. Verfügbar: http://aws.amazon.com/de/ec2/. [Besucht: 01-Aug-2015]
(Android DB) Adroid.com. „android.database.sqlite|Android Developers". [Online]. Verfügbar: http://developer.android.com/reference/android/database/sqlite/package-summary.html [Besucht: 01-Aug-2015]
(App Usability) eresult T. Wilhelm. „App-Usability – Herausforderungen und Guidelines". [Online]. Verfügbar: http://www.eresult.de/ux-wissen/forschungsbeitraege/einzelansicht/news/app-usability-herausforderungen-und-guidelines/ [Besucht: 01-Aug-2015]
(Apple Developer) Apple Inc.. „Which Developer Program is for you?". [Online] Verfügbar: https://developer.apple.com/programs/which-program/ [Besucht: 01-Aug-2015]
(CDYNE WetterWS) CDYNE Corporation. „Weather Web Service". [Online] Verfügbar: http://wsf.cdyne.com/WeatherWS/Weather.asmx?op=GetCityForecastByZIP [Besucht: 01-Aug-2015]
(chip Bildschirmauflösung) CHIP Digital GmbH „Brillante Handy-Displays: Die schärfsten Smartphones". [Online]. Verfügbar: http://www.chip.de/bildergalerie/Brillante-Handy-Displays-Die-schaerfsten-Smartphones-Galerie_58475373.html [Besucht: 01-Aug-2015]
(Computerwoche iPhone) Computerwoche.de IDG Business Media GmbH. „iPhone: Anwendung ohne App Store und Jailbreak installieren". [Online]. Verfügbar: http://www.computerwoche.de/netzwerke/mobile-wireless/1873768/. [Besucht: 01-Aug-2015]
(Conceptual Modeling) John Mylopoulos „Metamodeling". [Online]. Verfügbar: http://www.cs.toronto.edu/~jm/2507S/Notes04/Meta.pdf. [Besucht: 01-Aug-2015]
(derStandard Musik) derStandard.at GmbH „Apple kauft Kultkopfhörermarke Beats für drei Milliarden Dollar". [Online]. Verfügbar: http://derstandard.at/2000001642658/Apple-kauft-Kultkopfhoerermarke-Beats-fuer-drei-Milliarden-Dollar. [Besucht: 01-Aug-2015]
(derStandard US-Markt Smartphones) derStandard.at GmbH. „US-Markt - Android und iOS wachsen, Blackberry schrumpft, WP7 bleibt klein". [Online]. Verfügbar: http://derstandard.at/1313024867936/US-Markt-Android-und-iOS-wachsen-Blackberry-schrumpft-WP7-bleibt-klein. [Besucht: 01-Aug-2015]
(eclipse Framework) The Eclipse Foundation „Eclipse Luna". [Online]. Avaliable: http://eclipse.org/. [Besucht: 01-Aug-2015]
(EDGE) Teltarif. „Die Technik im Detail: So funktioniert EDGE". [Online]. Verfügbar:

http://www.teltarif.de/mobilfunk/edge/technik.html [Besucht: 01-Aug-2015]
(Galileo Serialisierung) Galileo Press. „Galileo Computing: Apps programmieren für iPhone und iPad". [Online]. Verfügbar: http://openbook.galileocomputing.de/apps_programmieren_fuer_iphone_und_ipad/1915_08_001.html [Besucht: 01-Aug-2015]
(Gartner Forecast Apps) Gartner, Inc. „Gartner Says Mobile App Stores Will See Annual Downloads Reach 102 Billion in 2013 ". [Online]. Verfügbar: http://www.gartner.com/newsroom/id/2592315. [Besucht: 01-Aug-2015]
(Gartner Revenue Apps) Gartner, Inc. „Forecast: Mobile App Stores, Worldwide, 2013 Update". [Online]. Verfügbar: https://www.gartner.com/doc/2584918 [Besucht: 01-Aug-2015]
(Gartner Smartphones 2010-2015) Gartner, Inc. „Android to Command Nearly Half of Worldwide Smartphone Operating System Market by Year-End 2012". [Online]. Verfügbar: http://www.gartner.com/it/page.jsp?id=1622614. [Besucht: 01-Aug-2015]
(GPRS) Teltarif. „Der General Packet Radio Service - kurz GPRS". [Online]. Verfügbar: http://www.teltarif.de/mobilfunk/gprs/ [Besucht: 01-Aug-2015]
(GSM Erfolgsgeschichte) Teltarif. „GSM – eine Erfolgsgeschichte". [Online]. Verfügbar: http://www.teltarif.de/mobilfunk/gsm.html [Besucht: 01-Aug-2015]
(Guardian Revenue Apps) Guardian News and Media Limited. „ Mobile apps revenues tipped to reach $26bn in 2013". [Online]. Verfügbar: http://www.theguardian.com/technology/appsblog/2013/sep/19/gartner-mobile-apps-revenues-report [Besucht: 01-Aug-2015]
(Heise.de 4K) Heise Zeitschriften Verlag. „Displays mit 4K-Auflösung in der Praxis". [Online]. Verfügbar: http://www.heise.de/ct/artikel/Gute-Aussicht-1711453.html [Besucht: 01-Aug-2015]
(Heise.de Handy-Verkaufszahlen) Heise Zeitschriften Verlag Online. „Handy-Verkaufszahrlen rückläufig, Smartphones im Aufwind". [Online]. Verfügbar: http://www.heise.de/newsticker/meldung/Handy-Verkaufszahlen-ruecklaeufig-Smartphones-im-Aufwind-750965.html. [Besucht: 01-Aug-2015]
(iOS DB) Apple Inc.. „Data Management - iOS Technology Overview - Apple Developer ". [Online]. Verfügbar: https://developer.apple.com/technologies/ios/data-management.html [Besucht: 01-Aug-2015]
(JSON Fat Free) Douglas Crockford, ECMA-404 The JSON Data Interchange Standard. „JSON". [Online] Verfügbar: http://www.json.org/fatfree.html [Besucht: 01-Aug-2015]
(JSON XML) Douglas Crockford, ECMA-404 The JSON Data Interchange Standard. „JSON: The Fat-Free Alternative to XML". [Online] Verfügbar: 2014]http://www.json.org/xml.html [Besucht: 01-Aug-2015]
(ksoap Beispiel) Thiranjith Weerasinghe. „Introduction to working with kSOAP2 in Android". [Online] Verfügbar: http://thiranjith.com/2012/10/15/introduction-to-working-with-ksoap2-in-android/ [Besucht: 01-Aug-2015]
(Lenovo kauft Motorola) Heise Medien Gruppe GmbH & Co. KG. „Google bestätigt: Lenovo kauft Motorola".[Online] Verfügbar: http://www.heise.de/newsticker/meldung/Google-bestaetigt-Lenovo-kauft-Motorola-2100811.html [Besucht: 01-Aug-2015]
(LTE Technik) Teltarif. „LTE Advanced setzt ausgefeilte Techniken ein". [Online]. Verfügbar: http://www.teltarif.de/mobilfunk/lte/advanced-technik.html [Besucht: 01-Aug-2015]
(Microsoft Patente) Microsoft Corporation. „Microsoft's New Patent Agreement with Compal: A New Milestone for Our Android Licensing Program". [Online] Verfügbar: http://blogs.technet.com/b/microsoft_on_the_issues/archive/2011/10/23/microsoft-s-

new-patent-agreement-with-compal-a-new-milestone-for-our-android-licensing-program.aspx [Besucht: 01-Aug-2015]
(Netztest AT) connect (WEKA MEDIA PUBLISHING GmbH). „WER HAT DAS BESTE HANDY-NETZ IN ÖSTERREICH?". [Online]. Verfügbar: http://www.connect.de/vergleichstest/bestes-handynetz-oesterreich-connect-netztest-2013-vergleich-a1-telekom-austria-drei-t-mobile-handy-netz-1900231.html [Besucht: 01-Aug-2015]
(Oracle JDK) Oracle Java Technology. „JDK Development Tools". [Online]. Verfügbar: http://docs.oracle.com/javase/6/docs/technotes/tools/. [Besucht: 01-Aug-2015]
(Open Models Resources) Open Models Initiative, „Open Models Resources". [Online]. Verfügbar: http://www.openmodels.at/resources. [Besucht: 1-Dec-2014]
(Open Source Definition) Open Source Initiative, „The Open Source Definition". [Online]. Verfügbar: http://www.opensource.org/docs/osd. [Besucht: 01-Aug-2015]
(Open Source Initiative) Open Source Initiative, „About the Open Source Initiative". [Online]. Verfügbar: http://www.opensource.org/about. [Besucht: 01-Aug-2015]
(POJO) Joe Littlejohn „jsonschema2pojo" [Online]. Verfügbar: http://www.jsonschema2pojo.org/. [Besucht: 01-Aug-2015]
(Smartphone Sales 2014) Gartner, Inc. „ Gartner Says Annual Smartphone Sales Surpassed Sales of Feature Phones for the First Time in 2013". [Online]. Verfügbar: http://www.gartner.com/newsroom/id/2665715 [Besucht: 01-Aug-2015]
(SOAP Body) W3C School. „SOAP Body Element". [Onine]. Verfügbar: http://www.w3schools.com/soap/soap_body.asp. [Besucht: 01-Aug-2015]
(SOAP Envelope) W3C School. „SOAP Envelope Element". [Onine]. Verfügbar: http://www.w3schools.com/soap/soap_envelope.asp. [Besucht: 01-Aug-2015]
(SOAP Header) W3C School. „SOAP Header Element". [Onine]. Verfügbar: http://www.w3schools.com/soap/soap_header.asp. [Besucht: 01-Aug-2015]
(Sourceforge kSOAP2) S. Haustein, J. Seigel. „kSOAP2". [Online]. Verfügbar: http://ksoap2.sourceforge.net/. [Besucht: 01-Aug-2015]
(Stanford Models Science) Stanford. „Models in Science (Stanford Encyclopedia of Philosophy)". [Online]. Verfügbar: http://plato.stanford.edu/entries/models-science/ [Besucht: 01-Aug-2015]
(SUN WSImport) SUN Microsystems. „JAX-WS 2.0 ea3 -- wsimport – Project Kenia". [Online]. Verfügbar: https://jax-ws.java.net/jax-ws-ea3/docs/wsimport.html. [Besucht: 01-Aug-2015]
(TecChannel Windows Phone 7) TecChannel IDG Business Media GmbH. „Windows Phone 7 im Unternehmen". [Online]. Verfügbar: http://www.tecchannel.de/kommunikation/handy_pda/2036982/windows_phone_7_im_unternehmen_bereitstellen/index6.html. [Besucht: 01-Aug-2015]
(UDDI InfoWorld) InfoWorld. „Microsoft, IBM, SAP discontinue UDDI registry effort". [Online]. Verfügbar: http://weblog.infoworld.com/realworldsoa/archives/2005/12/uddi_is_a_dead.html [Besucht: 01-Aug-2015]
(UDDI ZDNet) ZDNet - CBS Interactive. „IBM acknowledges bypassing UDDI; calls for new SOA registry standard". [Online]. Verfügbar: http://www.zdnet.com/blog/service-oriented/ibm-acknowledges-bypassing-uddi-calls-for-new-soa-registry-standard/864 [Besucht: 01-Aug-2015]
(W3C Validator) W3C. „XML Validator". [Online]. Verfügbar: http://www.w3schools.com/xml/xml_validator.asp. [Besucht: 01-Aug-2015]
(W3C Web Service) W3C. „Web Services Glossary". [Online]. Verfügbar: http://www.w3.org/TR/ws-gloss/. [Acessed: 01-Aug-2015]

(W3C WSDL 1.1) W3C. „Web Services Description Language (WSDL) 1.1“. [Online]. Verfügbar: http://www.w3.org/TR/wsdl.html [Besucht: 01-Dez-201]
(W3C WSDL 2.0) W3C. „Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language“. [Online]. Verfügbar: http://www.w3.org/TR/wsdl20/ [Besucht: 01-Aug-2015]
(W3C XML) W3C. „Extensible Markup Language (XML)“. [Online]. Verfügbar: http://www.w3.org/XML/. [Besucht: 01-Aug-2015]
(W3C XPath) W3C. „XML Path Language (XPath) 2.0 (Second Edition)“. [Online]. Verfügbar: http://www.w3.org/TR/xpath20/. [Besucht: 01-Aug-2015]
(Windows Phone DB) msdn Microsoft. „Local database for Windows Phone 8 “. [Online]. Verfügbar: http://msdn.microsoft.com/en-us/library/windows/apps/hh202860(v=vs.105).aspx [Besucht: 01-Aug-2015]
(Wirtschaftslexikon Online Smartphone) Gabler Verlag (Herausgeber). „Gabler Wirtschaftslexikon, Stichwort: Smartphone“. [Online]. Verfügbar: http://wirtschaftslexikon.gabler.de/Archiv/569824/smartphone-v1.html. [Besucht: 01-Aug-2015]
(Yahoo Weather WS) Yahoo! Inc.. „Yahoo! Web Services Example: search API via GET”. [Online] Verfügbar: http://developer.yahoo.com/java/samples/YahooWebServiceGet.java [Besucht: 01-Aug-2015]
(Yahoo WS Update) Yahoo! Inc.. „Important API Updates and Changes | ydnfourblog - Yahoo”. [Online] Verfügbar: http://developer.yahoo.com/blogs/ydn/important-api-updates-changes-8060.html [Besucht: 01-Aug-2015]

(HTC Magic) SoMobile. „HTC Magic gets surprise update to Android 2.2” [Online]. Verfügbar: http://www.somobile.co.uk/news/wp-content/uploads/HTC-Magic-on-Vodafone.jpg/ [Besucht: 01-Aug-2015]
(MS Patente Grafik) Microsoft Corporation. „Microsoft’s New Patent Agreement with Compal: A New Milestone for Our Android Licensing Program”. [Online] Verfügbar: http://mscorp.blob.core.windows.net/mscorpmedia/2011/10/7624.androidpatent.jpg [Besucht: 01-Aug-2015]]

11. Abbildungsverzeichnis

12. Tabellenverzeichnis

13. Anhang

13.1. Alternative zu einem Webservice

Man muss sich zuerst die Struktur der HTML-Seite ansehen, herausfinden, in welchem HTML-Tag die Information gespeichert wird, und kann dann, falls es eine XHTML/HTML valide Seite ist, mithilfe von XPath[46] den Inhalt des HTML-Tags abfragen. Falls die Seite nicht XHTML/HTML valide ist, muss man hier mit so genannten regular expressions (deutsch: Regulärer Ausdruck)[47] arbeiten und so die komplette HTML-Seite parsen. Den Aufwand für eine solche Abfrage sieht man wie folgt:

```
String stadt = "Wien"
String einheit = "Celsius"
String mytemperatur =""; String line = ""; String myresult =""; String mytemp = "";
URL url = new URL("http://de.wetter.yahoo.com/q?s="+stadt+"e="+einheit);
HttpURLConnection httpcon = (HttpURLConnection)url.openConnection();
httpcon.setRequestMethod("GET");
httpcon.setUseCaches(false);
BufferedReader URLinput = new BufferedReader(new
InputStreamReader(httpcon.getInputStream()));
while ((line = URLinput.readLine()) != null) { myresult = myresult + line; }
Pattern mypattern = Pattern.compile("<wetter>Letztes Wetter:.*\\d++,\\d++.*</wetter>");
Matcher mymatch = mypattern.matcher(myresult);
while ( mymatch.find() ) {
   mytemp = mytemp + myresult.substring(mymatch.start(), mymatch.end());
}
mypattern = Pattern.compile("\\d++,\\d++");
mymatch = mypattern.matcher(mytemp);
while ( mymatch.find() ) {
   mytemperatur = mytemperatur + mytemp.substring(mymatch.start(), mymatch.end());
}
```

Auch hier liefert der Programmcode das aktuelle Wetter in der Einheit „Celsius“ für die ausgewählte Stadt „Wien“. In der Variablen „mytemperatur“ in der vorletzten Zeile wird aus einem Teil der kompletten Homepage dieser Wert herausgefiltert. Man sieht hier

[46] XPath ist eine Abfragesprache, die als Recommendation vom W3C im Jahre 1999 zum ersten Mal veröffentlicht wurde. Die aktuelle Version ist 2.0 und seit 2010 gültig. Diese Sprache ermöglicht es gezielte Inhalte aus einer XML- oder auch XHTML-Seite herauszufiltern bzw. zu adressieren. Man kann nicht nur Attribute oder Inhalte von speziellen XML-Tags ermitteln sondern auch einige Standardfunktionen verwenden, wie etwa eine Zählfunktion oder eine Textausgabefunktion. (Einstieg in XML) (W3C XPath)

[47] Ein Regulärer-Ausdruck ist eine bestimmte Anordnung von Zeichen, die man in einem Dokument sucht. Dieser Ausdruck muss nicht immer eindeutig sein, sondern kann auch Platzhalter, Leerzeichen, Zahlen, Buchstaben oder beliebige Zeichen enthalten. In dem Programmcode kommt das Beispiel: „<wetter>Letztes Wetter:.*\\d++,“ vor. Dieser Ausdruck sucht nach einer Zeichenfolge, die mit „<wetter>Letztes Wetter:“ beginnt, danach dürfen beliebige Zeichen folgen „.*“. Darauf erwartet man beliebig viele Zahlen „\\d“ einen Beistrich „++,“. (Einstieg in XML) (Wirtschaftsinformatik I)

sofort, dass der Programmieraufwand um einiges größer ist. Ein weiterer großer Nachteil ist, wenn der Eigentümer der Seite diese verändert, muss man das Programm anpassen. Dieses Beispiel soll zeigen, dass Webservices von großem Nutzen für den Datenaustausch im Internet sind. Informationen über Modelle können so viel einfacher von einer Maschine zu einer anderen transportiert werden.

13.2. Datenbankanforderungen

Hinzufügen von Daten:
Die Datenbank muss neue Datensätze abspeichern können.

```
INSERT INTO Modell-Tabelle VALUES ('1', 'Wetter', '20140723083045', '{"Wetter"}');
```

Entfernen von Daten:
Das Entfernen von einzelnen Datensätzen wird benötigt, damit die SQLite Datenbank nicht unnötig Speicherplatz mit alten Daten belegt.

```
DELETE FROM Modell-Tabelle WHERE ID = '1';
```

Auslesen von Daten:
Das Auslesen von Daten kann mit einem einfach „Select“-Befehl durchgeführt werden.

```
SELECT * FROM Modell-Tabelle WHERE ID = '1';
```

Zählen von Tabelleneinträgen:
Durch Möglichkeit des Customizings für den Benutzer muss die Datenbank Einträge zählen können.

```
SELECT COUNT(*) FROM Modell-Tabelle;
```

13.3. Erstellen einer Tabelle mit Android

Beim erstmaligen Installieren der Applikation in einer Android Umgebung werden drei Datenbanktabellen in Form von SQLite Tabellen angelegt:

```
private static final String CREATE_MODELL_TABELLE =
" create table Modell-Tabelle ( ID integer primary key autoincrement," +
" Name text primary key not null, Datum text not null, JSON text not null);";
```

```
private static final String CREATE_LOG_TABELLE =
" create table Log-Tabelle ( ID_LOG integer primary key autoincrement," +
" Datum text not null, Webservice text, EigenesModell text, HochgeladenesModell text, JSON text not null, foreign key (ID) References Modell-Tabelle(ID), foreign key (Name) References Modell-Tabelle(Name));";

private static final String CREATE_ON_DEMAND_TABELLE =
" create table On-demand-Tabelle ( ID_DEMAND integer primary key autoincrement," +
" OnDemandName text primary key, Datum text primary key, JSON text not null, foreign key (ID) References Modell-Tabelle(ID, foreign key (Name) References Modell-Tabelle(Name));";
```

In den String Variablen (CREATE_MODELL_TABELLE, CREATE_ON_DEMAND_TABELLE und CREATE_LOG_TABELLE) wird der Befehl zum Anlegen der Datenbanken inklusiver aller Attribute, Datentypen und Einschränkungen gespeichert. Folgender Code zeigt eine Methode, um diese Datenbanken zu erzeugen:

```
@Override

public void onCreate(SQLiteDatabase database) {
  database.execSQL(CREATE_MODELL_TABELLE);
  database.execSQL(CREATE_LOG_TABELLE);
  database.execSQL(CREATE_ON_DEMAND_TABELLE);
}
```

14. Anhang: Abstract

This thesis deals with the topic modeling on mobile devices. In the area of informatics models are used in different fields, like software engineering or business process modeling. In the introduction the author described that it is hard for a user to become an overview of a model on a small screen. A human being can hardly recognize more than five objects after a glimpse. This is the reason why models should be optimized for small devices. The user should concentrate on important parts of a model and after getting an overview the user should be confronted with more details of the model. Beside of reducing information there is also the possibility of coloring elements and relationships, focus on essential parts of the model or transform models into text. Different modeling languages need different methods of adaption. In chapter 3.2.1 the author analyses the class diagram and comes to the conclusion that an information reduction or centering on parts of the model. With reducing information the important elements and relationships move in the center of attention. Less important information like attributes or methods are hidden. With clicking on classes these hidden information will show up for the user. To center the view on, for the user, interesting parts of the model can support the information reduction even more. To center the view on a class will hide all other classes which are not related to this selected class. In chapter 3.2.2 the business process model notation is analyzed. This is a very complex modeling language which offers a user more than 25 different elements and relationships. Models can get really big and confusing so the transformation of the model into text seems to be the best optimization. In chapter 3.2.3 the use case diagram is analyzed. For this kind of model the centering view looks like the best optimization method. Furthermore also coloring parts of the model can be a suitable solution.

Models are transferred via web service to a smartphone. The recommended technology is REST because data can be transferred faster than with other techniques. The chosen data standard is JSON because data traffic can be slim and small data packages can be used. This is also a big advantage for the battery power since less data packages have to be transferred to the smartphone. Another big advantage is the possibility to send smaller data packages with slower networks and still receiving acceptable results in transferring the data.

In chapter 5 a data structure is defined to cover up all requirements which the application should handle. A connection loss can be handled with a logging table. This table can also be used if the application crashes, the battery is empty or the user forgets to save. To reduce data traffic model versions are saved on the smartphone and if no new version is available no data is loaded. The data structure helps with the optimization of reducing the information, centering the view or transforming the model into text.

The last section of this thesis contains general requirements for a smartphone application. These requirements handle technical as well as user interface aspects.

15. Anhang: Lebenslauf

Markus Köhler, Bakk
+43 664 8121115
a0451600@unet.univie.ac.at

Berufserfahrung

Zeitraum:	von Dezember 2011 bis heute
Name des Arbeitgeber:	Wienerberger AG
Position:	SAP CRM Application Consultant
Zeitraum:	von August 2009 bis August 2010
Name des Arbeitgeber:	Informations-Technologie Austria GmbH
Position:	System-Architekt
Zeitraum:	von März 2007 bis Juli 2010
Name des Arbeitgeber:	Universität Wien
Position:	Tutor für diverse Lehrveranstaltungen
Tätigkeit:	Vorbereitung von Lehrveranstaltungen & Prüfungen, Korrektur von wissenschaftlichen Arbeiten & Projekten, Unterstützung der Studenten
Zeitraum:	von Februar 2008 bis November 2009
Name des Arbeitgeber:	Universität Wien
Position:	Projektmitarbeiter
Tätigkeit:	Evaluierungen für das Projekt eduBITE, Video-Tutorials für eduWeaver

Praktika

Zeitraum:	Juli 2007
Name des Arbeitgeber:	PORR AG
Position:	Praktikant beim technischen Innendienst
Zeitraum:	August 2006
Name des Arbeitgeber:	ARGE Tunnel Wienerwald
Position:	Assistent des Baukaufmanns
Zeitraum:	von Juni bis August 2004 & von Juli bis August 2005
Name des Arbeitgeber:	ARGE WSKE-Wientalsammler
Position:	Assistent der Bauleitung
Zeitraum:	Juli 2000 & Juli 2001
Name des Arbeitgeber:	PORR AG
Position:	Praktikant in der Zentralwerkstadt

Ausbildung

Zeitraum:	2009-heute
Universität:	Masterstudium Wirtschaftsinformatik (Universität-Wien)
Masterarbeit	„Modellierung mit mobilen Devices: Ein Technologievergleich und eine Applikationsbewertung“
Zeitraum:	08.2010-01.2011

Universität:	ERASMUS Masterstudium Wirtschaftsinformatik (Universität-Stockholm)
Zeitraum:	2005-2009
Universität:	Bakkalaureatsstudium Wirtschaftsinformatik (Universität-Wien)
Abschluss:	Abschluss des Bakkalaureatsstudiums (Bakk. rer. soc. oec.)
Bakkalaureatsarbeit 1	„The Standard for Program Management"
Bakkalaureatsarbeit 2	„Hybrid Blog Recommender"
Zeitraum:	2004-2005
Universität:	Bakkalaureatsstudium Wirtschaftsinformatik (WU-Wien)

Zeitraum:	1999-2003
Schule:	BRG Wien 23 - Anton Baumgartnerstraße
Abschluss:	14.06.2003 Matura am BRG23
Zeitraum:	1995-1999
Schule:	BRG Mödling - Keimgasse

Fähigkeiten / Kenntnisse

Fremdsprachen

- Deutsch - Muttersprache
- Englisch - verhandlungssicher
- Französisch - Gute Kenntnisse
- Schwedisch - Grundkenntnisse
- Polnisch - Grundkenntnisse

EDV-Kenntnisse

- Office-Software - sehr gut
- Programmiersprachen: ABAP, ABAP Objects, Java, C++, SQL (Oracle, MS, MySQL), XML-Family, HTML, Javascript, - gut bis sehr gut
- Photoshop - gut
- Programmierte Homepages: www.hno-koehler.at & www.pvba.at

Auslandsaufenthalte

- 1999 - Sprachreise nach Malta mit EF
- 2000 - Sprachreise nach England mit Ökista
- 2001 - Sprachreise nach England mit SFA (Schule)
- 2010 - ERASMUS Aufenthalt an der Universität Stockholm

Wissenschaftliche Tätigkeiten

Zeitraum	23.02.2010 bis 25.02.2010
Tätigkeit	Teilnahme an der MKWI 2010 Göttingen
Zeitraum	25.02.2010 bis 27.02.2009
Tätigkeit	Mitarbeit und Teilnahme an der WI 2009 Wien
Zeitraum	26.02.2008 bis 28.02.2008
Tätigkeit	Teilnahme an der MKWI 2008 München
Zeitraum	Sommersemester 2009
Tätigkeit	Accenture Campus Challenge