TU Clausthal
Institut für Informatik

Bachelorarbeit

Anpassung von argumentationsunterstützenden Systemen an die Anforderungen mobiler Endgeräte am Beispiel des iPod touch

Benjamin Neu

11. Mai 2010

Bachelorarbeit

Anpassung von argumentationsunterstützenden Systemen an die Anforderungen mobiler Endgeräte am Beispiel des iPod touch

eingereicht bei

Betreuender Prüfer: Prof. Dr. Niels Pinkwart
Zweitgutachter: Prof. Dr. Gabriel Zachmann
Institut für Informatik
Abteilung für Wirtschaftsinformatik
Fakultät für Mathematik/Informatik und Maschinenbau
Technische Universität Clausthal

von
Benjamin Neu
Studienrichtung: Wirtschaftsinformatik, B. Sc.
Matrikelnummer: 354486

Datum: 11. Mai 2010

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

CSS	Cascading Style Sheets
DFKI	Deutschen Forschungszentrums für Künstliche Intelligenz
GWT	Google Web Toolkit
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDEs	integrierte Entwicklungsumgebungen
LASAD	Learning to Argue: Generalized Support Across Domains
SDKs	Software Development Kit
SOAP	Simple Object Access Protocol

1 Einleitung

1.1 Motivation

Eine nötige Alltagsfähigkeit des Menschen ist das Argumentieren. Immer wieder ist die eigene Meinung gegenüber anderen zu begründen oder es sind die Argumente eines anderen zu entkräften. Die zentrale Bedeutung dieser Fähigkeit führt dazu, dass sie in der Schule, im Studium sowie im Beruf vermittelt und weiter ausgebildet werden soll. Im Optimalfall wird die Fähigkeit durch eine 1:1 Relation zwischen Lehrendem und Schüler vermittelt. In der Realität ist dies aber aus verschiedensten Gründen nicht immer möglich. Um diesen Umstand auszugleichen, wurden argumentationsunterstüzende Systeme entwickelt, die ein individuelles individuelles Ausbilden der Argumentationsfähigkeit mit Hilfe der Computertechnologie fördern sollen.

Mobile Endgeräte in ihren verschiedenen Formen haben sich in den letzten Jahren zum ständigen Begleiter vieler Menschen entwickelt. Durch die zunehmend besser werdende Mobilfunktechnik hat man die Möglichkeit, nahezu von überall auf das Internet zuzugreifen und sich Daten und Software herunterzulanden. Aus diesem Grunde entstanden zahlreiche neue an mobile Endgeräte angepasste Softwareprodukte und für zahlreiche klassische Desktopanwendungen wurden mobil angepasste Versionen geschrieben.

Daher ist es naheliegend, für mobile Endgeräte vorhandene argumentationsunterstützende Systeme zu portieren, um praktisch jederzeit und überall Zugriff auf eine Argumentationshilfe zu haben.

1.2 Aufgabenstellung

Ziel dieser Arbeit ist zum Einen die Evaluation der nötigen Anpassungen argumentationsunterstützender Software hinsichtlich der Anforderungen und Einschränkungen

mobiler Plattformen. Folgende Aspekte müssen hierzu im Folgenden untersucht werden:

- Eignung vorhandener Benutzeroberflächen für mobile Anwendungen
- Funktionsumfang mobiler Anwendungen
- Datenaustausch zwischen Gerät und System

Zum Anderen werden die gewonnenen Erkenntnisse dann zur Entwicklung einer argumentationsstützenden Anwendung verwendet. Diese Anwendung wird auf dem bereits existierenden LASAD (siehe 2.1.2) System, welches an der TU-Clausthal mitentwickelt wird, beruhen. Als Zielplattform wird ein iPod touch zum Einsatz kommen.

2 State of the art

2.1 Argumentationsunterstüzende Systeme

2.1.1 Argumentationsunterstüzende Systeme

Argumentieren ist eine Alltagsfähigkeit, die privat und insbesondere beruflich von hoher Bedeutung ist. Es wurden sogenannte argumentationsunterstüzende Systeme entworfen, die entweder das Erlernen von Argumentationsfähigkeiten unterstützen oder aber als Werkzeug zur Unterstützung von Argumentationen dienen sollen. Im Rahmen des Review „ Computer-supported argumentation: A review of the state of the art “ ([4]) wurden 50 verschiedene solcher Systeme untersucht und nach verschiedenen Kriterien unterschieden. Unter anderem wurde hier nach folgenden Kriterien unterschieden:

- Erlernen steht im Fordergrund oder Unterstützung von Argumentationsprozessen
- Einzelarbeit oder kollaboratives Arbeiten
- Domänen spezifisches oder allgemeines System
- Art der Darstellung

Wie verschiedenen diese einzelnen Systeme sind, zeigt sich nicht nur an diesen Unterscheidungskriterien sondern auch an den zu Grunde liegenden Basistechnologien. Nach [1] und [4] sind die meisten Systeme individuell „from the scratch“ entwickelt worden und bedienen sich keiner gemeinsamen Methodik. Dies ist bemerkenswert, haben sich doch im Allgemeinen bei der Softwareentwicklung Entwicklungsmuster als praktikable Lösung zum Übernehmen von Lösungsstrategien erwiesen.

Ein Umstand, der dazu beiträgt, dass ein entwickeltes System vielfach nicht wieder verwendet wird, ist, dass es bei vielen Systemen nicht einfach möglich ist, die im Rahmen der Diskussion benötigten Elemente an eine andere Domäne anzupassen. Eine

Spezifikation der benutzbaren Elemente und ihrer Beziehungen untereinander nennt man „Ontology“ (vgl. [1] - 5 Ontologies). Je nach Einsatzgebiet ergeben sich so spezielle Ontologien, die das Abbilden von Argumentationen innerhalb des Einsatzgebietes ermöglichen. Bestehende Systeme sind meist auf ein Einsatzgebiet beschränkt und können nicht domänenübergreifend verwendet werden. Des Weiteren bieten die meisten bestehenden Systeme keine oder nur eingeschränkte Anpassungsmöglichkeiten an.

2.1.2 Das LASAD System

Mit der Entwicklung eines flexiblen argumentationsunterstützenden Systems beschäftigt sich das Projekt Learning to Argue: Generalized Support Across Domains (LASAD) des Deutschen Forschungszentrums für Künstliche Intelligenz (DFKI) und der Technischen Universität Clausthal. Auf der Projektseite ([5]) gibt es folgende Zieldefinition : „Our objective in project LASAD is to create a generalized framework and methodology for the construction of argumentation support systems to help students learn argumentation in different domains.“ Hauptziel ist es, ein wieder verwendbares Framework zu schreiben, das innerhalb verschiedenster Domänen zum Einsatz kommen kann. Insbesondere wird hierbei Wert auf den möglichst flexiblen Einsatz verschiedenster Ontologien gelegt. Neben der Offenheit gegenüber der Domäne ist das System im Allgemeinen auf eine loose coupling- Architektur ausgerichtet, so dass einzelne Teile des Systems mit geringem Aufwand getauscht werden können.

In Abbildung 2.1 ist die aktuelle Architektur aufgezeigt. Die Architektur entspricht einer 3 Schichten Architektur: Präsentation-, Logik- und Datenhaltungsschicht. In der Präsentationsschicht unterscheidet man zwischen zwei verschiedenen Arten von Clients. Zum einen gibt es den Userclient. Dieser bietet für den Anwender einen Arbeitszugang zum System an. Anforderungen an den Client waren unter anderem, dass keine Installation erforderlich ist und bei der Kommunikation zum Server keine Probleme durch eine eventuell vorhandene Firewall entstehen. In der Betaversion wurde der Userclient mit Hilfe des Google Web Toolkit (GWT) implementiert. Die zweite Art von Client sind Analyse- und Feedbackclients. Sie analysieren die vom Nutzer eingegebenen Argumente, können vielfältiges Feedback bezüglich der Argumente an den oder die Nutzer geben. In der Betaversion sollen sie als Java-Anwendungen realisiert werden, die mittels Webservice auf die Logikschicht zugreifen. Herzstück des Systems ist die

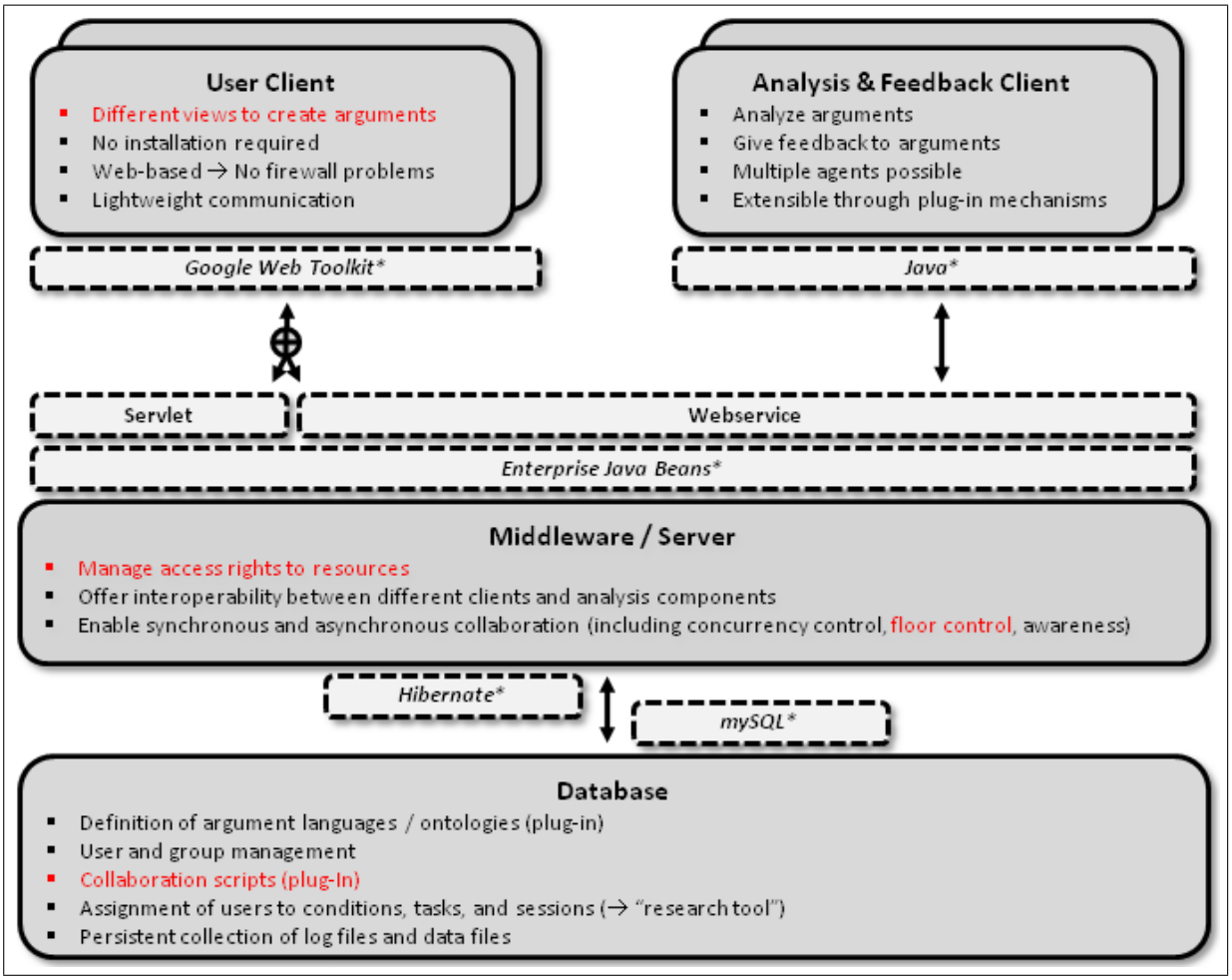

Abbildung 2.1: LASAD Systemarchitektur, Quelle: [1]

Logikschicht. Ein zentraler Server kümmert sich um die Rechteverwaltung und um die Lösung von Nebenläufigkeitsproblemen bei kooperativem Arbeiten. In der aktuellen Betaversion (Stand 6. April 2010) wird die Logik mit Hilfe von Enterprise JAVA Beans realisiert. Mittels Servlet und über einen Webservice bietet der Server Schnittstellen für die Präsentationsschicht an. Die Datenhaltungsschicht kümmert sich um die persistente Speicherung der Anwendungsdaten sowie die Ontologiedefinitionen. Aktuell wird dies mit Hilfe einer mySQL-Datenbank realisiert, auf die mittels Hibernate von der Logikschicht zugegriffen werden kann.

Zurzeit bildet das System bereits testweise das LARGO, Belvedere und Argonaut System nach. Alle 3 Systeme betreffen unterschiedliche Domänen und besitzen jeweils eine

individuelle Ontologie. Durch die exemplarische Umsetzung dieser drei zeigt das System bereits das Streben nach Offenheit gegenüber beliebigen Domänen.

2.2 Mobile Endgeräte

2.2.1 Mobile Endgeräte im Allgemeinen

Befasst man sich mit mobilen Endgeräten, so ist zunächst zu klären, welche Geräte unter diesen Begriff fallen. Nach Turowski und Pousttchi ([6] Kapitel 3) zeichnen sich mobile Endgeräte insbesondere durch folgende Eigenschaften aus:

- konzipiert für mobilen Einsatz
- 1:1 Relation zum Benutzer
- mobile Mehrwert der Allgegenwärtigkeit

Mit der Konzipierung für den mobilen Einsatz ist vor allen Dingen der Einsatz über einen gewissen Zeitraum ohne eine externe Stromversorgung gemeint. Eine 1:1 Relation zwischen Gerät und Benutzer ergibt sich aus der Benutzungsweise des Gerätes. So werden im Regelfall personenbezogene Daten gespeichert wie Kontakte und Termine oder aber, wie im Falle des Mobiltelefons, möchte man gezielt erreicht werden können. Durch die sehr gute Transportabilität des Gerätes und die Möglichkeit es während des Transportes jederzeit benutzten zu können, entsteht der mobile Mehrwert der Allgegenwärtigkeit. Es ist dem Benutzer jederzeit möglich auf seine Daten zuzugreifen oder mit anderen Personen zu kommunzieren, was er sonst wohlmöglich nur an einem stationären Endgerät könnte.

Zwar passen diese Eigenschaften auch auf ein Notebook, jedoch betrachten Turowski und Pousttchi dieses nicht als mobiles Endgerät, da man es im Normalfall nicht während des Transportes selbst benutzt.

Zur Zeit gibt es 3 große Kategorien von Mobilen Endgeräten ([6] Kapitel 3.3):

- Mobiltelefone jeglicher Generation
- Personal Digital Assistants (PDA)
- Smartphones

Die besondere Entwicklung des Smartphones mit steigender Verbreitung soll im nächsten Abschnitt hervorgehoben werden. Neben den Hauptkategorien gibt es noch weitere kleine Kategorien oder aber Spezialanfertigungen für besondere Einsatzzwecke. Erwähnen sollte man hier Ebook-Reader und Tablet-Computer. Tablet-Computer, um die es in den letzten Jahren eher ruhig war, erfahren durch die Markteinführung des iPads und vieler ähnlicher Produkte einen neuen Aufschwung und könnten sich bald ebenfalls zu einer Hauptkategorie entwickeln.

2.2.2 Smartphones

In den letzten 15 Jahren haben sich mobile Endgeräte einen festen Platz in unserem Alltag erobert. Angefangen hat dieser Siegeszug mit simplen PDAs und der ersten Generation von Mobiltelefonen. Daraus hat sich ein Hybrid entwickelt: das sogenannte Smartphone. Es vereint Funktionen herkömmlicher PDAs mit den Kommunikationsmöglichkeiten eines modernen Mobiltelefons.

Diese modernen Geräte besitzen nunmehr eine Rechenleistung, die vor einigen Jahren noch teuren Workstations vorbehalten war und es nun im Hosentaschenformat gibt. Hinzu kommen die Fortschritte in der mobilen drahtlosen Kommunikationtechnik, so dass mobile Endgeräte schon jetzt Datentransferraten vergleichbar mit DSL nutzen können, um auf die verschiedenste Dienste, wie zum Beispiel das Internet, zugreifen zu können.

An den Verkaufszahlen der Jahre 2008 und 2009 kann man erkennen, dass Smartphones dabei sind, die herkömmlichen Mobiltelefone abzulösen. Mit etwa 1,211 Milliarden verkauften Mobiltelefonen im Jahre 2009 wurden im Vergleich zum Vorjahr etwa 10 Millionen weniger verkauft. Demgegenüber ist der Absatz von Smartphones um 30 Millionen auf etwa 170 Millionen Stück im Jahr 2009 gestiegen (vgl. [7] und [8]).

Dieser weltweite Trend ist auch in Deutschland zu sehen. Es wurden 2008 3,1 Millionen und 2009 5,6 Millionen Smartphones in Deutschland verkauft. Für das Jahr 2010 geht der Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V. von einem Absatz von 8,2 Millionen Smartphones in Deutschland aus, was bedeuten würde, dass jedes 3. verkaufte Handy ein Smartphone wäre (vgl. [9]).

Ein Aspekt, nach dem sich Smartphones unterscheiden lassen, sind die Betriebssysteme.

BS Hersteller	Marktanteil 2009	Marktanteil 2008	Veränderung
Symbian	46.9 %	52.4 %	-5.5%
Research in Motion	19.9 %	16.6 %	+3.3%
Apple	14.4 %	8.2 %	+4.2%
Microsoft	8.7 %	11.8 %	-3.1%
Google	3.9 %	0.5 %	+3.4%
Andere	6.2 %	10.5 %	-4.3%

Tabelle 2.1: Smartphone Marktübersicht nach Betriebssystem Herstellern

Eine Tabelle (siehe Tabelle 2.1) der Gartner Inc. ([7]) listet die großen Betriebssystemhersteller mit ihren Marktanteilen der letzten beiden Jahre auf. An der Entwicklung der Marktanteile von 2008 zu 2009 lässt sich ein weiterer Trend innerhalb der Smartphones erkennen. Marktanteile verloren haben unter anderem Symbian mit -5.5% und Microsoft mit -3.1%, welche beide länger am Markt sind. Verbessern konnten sich Apple mit +4.4% und Google mit + 3.4%. Beide sind im Vergleich zu Symbian oder Microsoft Neulinge am Smartphone Markt. Der schwerpunktmässige Einsatz der Betriebssysteme von Apple und Google ist auf Geräten mit Touch-Screens. Genau diese Geräte verzeichneten von 2008 auf 2009 eine Erhöhung des Absatzes um ca. 40 Millionen auf 75 Millionen Stück im Jahr 2009 (vgl. [8]). Offen bleibt hier die Frage, ob Betriebssystem oder Touch-Screen zu erhöhtem Absatz führten.

2.2.3 Der iPod touch

Apple beschreibt den iPod auf die Frage „Was ist der iPod touch ?“ mit den Worten

> „Ein großartiger iPod. Ein großartiger Taschencomputer. Großartiger mobiler Spielspaß.“ ([10])

Betrachtet man auszugsweise die offizielle Hardware Spezifikation ([11]) erkennt man einen etwa handflächengroßen 8,5mm dicken mobilen Computer. Aufgrund seiner „touch“ Bedienung kann fast die gesamte Oberfläche des Gerätes durch den Bildschirm eingenommen werden und so der beschränkte Platz bestmöglich ausgenutzt werden. 3 Knöpfe besitzt der iPod trotzdem: einen An- und Ausschalter an der Oberseite , einen Lautstärkeregler an der Seite und einen „Homebutton“ unterhalb des Displays, über den man immer wieder in das Hauptmenü zurückkehren kann.

Größe	Höhe 110mm Breite 61.8 mm Tiefe 8.5 mm
Gewicht	115 Gramm
Bildschirm	3,5“ Widescreen–Multi-Touch–Display
Bildschirmauflösung	480 x 320 Pixel
Speicherkapazität	32 GB oder 64 GB Flash–Laufwerk
Funkunterstützung	Wi-Fi (802.11b/g) Bluetooth 2.1 + EDR
Ein- und Ausgänge	Dock-Anschluss Stereo-Kopfhöhreranschluss

Tabelle 2.2: Auszug aus der iPod touch Spezifikation

Der iPod touch wird mit iPhone OS als Betriebssystem ausgeliefert und kommt mit einer Auswahl an Standartsoftware aus den folgenden Bereichen:

- Multimedia (Bilder, Musik, Video)
- Browser
- Email
- Kalender
- Adressbuch

Man hat also direkt nach Auslieferung den Funktionsumfang eines PDA kombiniert mit den Multimediamöglichkeiten zum Abspielen von Bildern, Musik und Videos. Dieser Funktionsumfang ähnelt dem moderner Smartphones. Aufgrund des fehlenden Telekommunikationsmoduls gehört er laut Kategorisierung von 2.2.1 in die Kategorie der PDAs.

Der Softwareumfang des Gerätes lässt sich mit so genannten Apps leicht erweitern. Bezogen wird diese Software über eine Internetplattform von Apple, dem App-Store. Im App-Store sind zur Zeit etwa 185.000 Apps eingetragen, die teils kostenlos teils kostenpflichtig herunterladen und auf dem Gerät ausgeführt werden können. Apps werden hierbei nicht nur von Apple sondern auch von einer Vielzahl externer Anbieter erstellt und über den App-Store vertrieben.

Herausgestellt werden muss an dieser Stelle, dass der Massenvertrieb von Software für iPhone OS nur über den App-Store möglich ist. Apple behält sich so die Kontrolle über den Markt vor und verdient über eine Verkaufsprovision an jedem einzelnen Verkauf mit. Um eine Anwendung in den App-Store einzustellen, muss diese zunächst einem Prüfverfahren unterzogen werden und kommt erst nach Genehmigung von Apple zum Verkauf. Diese Zensurmöglichkeit wird in der Fachwelt vielfach diskutiert.

Entwickelt werden die Applikationen mit dem von Apple bereitgestellten iPhone SDK. Mit dem iPhone SDK entwickelte Anwendungen laufen in den meisten Fällen sowohl auf dem iPod touch wie auch auf dem iPhone und dem iPad. Ausgenommen sind zum Beispiel Anwendungen welche telekommunikationsspezifische Elemente beinhalten, welche dem iPhone vorbehalten sind.

Weltweit wurden bis April 2010 85 Millionen iPhone und iPod touch verkauft. Diese verteilen sich auf 50 Millionen iPhones und 35 Millionen iPod Touch ([12]).

3 Design/Entwurf

3.1 Besondernheiten bei Softwareentwicklung für mobile Endgeräte

Beim Entwurf von Software für mobile Endgeräte muss man sich zunächst die Besonderheiten gegenüber der Softwareentwicklung für Desktoprechner bewusst sein. Es gibt zahlreiche Einschränkungen bezüglich der eingesetzten Hard- und Software, aber auch das Nutzerverhalten unterscheidet sich stark von dem eines Desktopbenutzers. Im Folgenden soll auf diese Punkte genauer eingegangen werden.

3.1.1 Einschränkungen durch mobile Hardware und mobile Betriebssysteme

Betrachten wir zunächst die Hardware-Einschränkungen von mobilen Endgeräten. Diese können im Wesentlichen auf die folgenden Punkte zusammengefasst werden (vgl. [13] S. 243f, [14] S. 118f und [15] S. 15f):

- begrenzte Akkulaufzeit
- beschränkte CPU Rechenleistung
- geringer Arbeitsspeicher
- Bildschirmgröße und Auflösung
- Art der Eingabemöglichkeit
- Beschränkte Bandbreite für Datenübertragung

Auch mit modernster Akkutechnologien erreichen mobile Endgeräte unter Auslastung nur eine geringe Betriebszeit von wenigen Stunden ohne externe Stromversorgung. Die durch bessere Akkus in den letzten Jahren bereitgestellte Zusatzkapazität wurde durch

Entwicklungen wie kleine Farbdisplays, GPS-Module, WLAN- und UMTS- Module zeitgleich neutralisiert.

Durch die beschränkte Akkuleistung und die Größe von mobilen Endgeräten ist selbst bei aktuellsten Geräten im Vergleich zu Desktopcomputer nur eine stark begrenzte Rechenleistung vorhanden. Die Ursache dafür ist, dass viel Leistung zurzeit auch einen hohen Energieverlust bedeutet aber ein ausgewogenes Verhältnis zwischen Laufzeit und Leistung erreicht werden soll.

Kann man heutzutage fast verschwenderisch mit dem Arbeitsspeicher auf einem Desktoprechner umgehen, so muss man bei mobilen Endgeräten versuchen, möglichst wenig Speicher zu verbrauchen. Gegenüber dem Desktop-PC ist nur verhältnismässig wenig Arbeitsspeicher verbaut, der bei Unachtsamkeit schnell komplett ausgenutzt ist. Verbraucht man zum Beispiel auf iPhone OS Geräten zuviel Arbeitsspeicher, kann es passieren, dass die Anwendung komplett abgebrochen wird.

Eine sehr starke Einschränkung für die mobile Anwendungsentwicklung ist der Bildschirm des Gerätes. Im Normalfall ist bei einem mobilen Endgerät ein weitaus kleinerer Bildschirm mit geringerer Auflösung vorzufinden als bei einem Desktop Rechner. Aus diesem Grunde muss gut überlegt werden, wie die gewünschten Informationen / Daten auf den Bildschirm dargestellt werden ohne dass der Nutzer den Überblick verliert.

Die vorhandenen Eingabemöglichkeiten können sehr unterschiedlich sein. Als Hauptarten sind eine Mini-Tastatur, Toucheingaben oder sprachgesteuerte Eingaben zu nennen. Unabhängig davon, welche diese Eingabemöglichkeiten vorhanden sind, kann man davon ausgehen, dass der Benutzer zum einen langsamer bei der Eingabe ist und vor allen Dingen weniger Informationen eingeben wird, als er dies über eine übliche Computertastatur tun würde. Auf diesen Umstand wird im nächsten Abschnitt 3.1.2 noch näher eingegangen.

Zur Datenübertragung stehen bei mobilen Endgeräten verschiedenste Technologien zur Verfügung. Man unterscheidet hierbei die drei Gruppen Wireless-WAN-Technologien, Wireless-LAN-Technologien und Wired-Technologien (vgl. [16]). Wireless-LAN und Wired-Technologien liefern örtlich begrenzt gute Übertragungsraten. Demgegenüber ist die allgemeine Netzabdeckung durch Wireless-WAN-Technologien bisher nicht immer und überall gegeben. Aus diesem Grunde sollte man mobile Applikationen auf geringe Datenübertragungsvolumen hin auslegen.

Für mobile Endgeräte gibt es eine Vielzahl von mobilen Betriebsystemen, durch die teilweise wiederum Einschränkungen bei der Entwickelung der mobilen Anwendung gegeben sind. So ist zum Beispiel Art und Umfang der einsetzbaren Programmiersprachen bei den Betriebssystemen sehr unterschiedlich. Hat man bei Android eine große Auswahl an Programmiersprachen so beschränkt sich diese beim iPhone OS üblicherweise auf Objective-C oder C. Die Beschränkung kann sowohl gute als auch schlechte Effekte mit sich bringen. Zum Beispiel bringt die Entwicklungsumgebung von iPhoneOS, welches auf Objective-C ausgerichtet ist, eine Vielzahl von Standardelementen für die Gestaltung der Benutzeroberfläche mit. Auf der anderen Seite hat sie jedoch an manchen Stellen einen eingeschränkten Funktionsumfang, so fehlt zum Beispiel eine Implementierung des Simple Object Access Protocol (SOAP)). Andere Betriebssystemaspekte, auf die nicht näher eingegangen werden soll, sind unter anderem der Ressourcenbedarf des Betriebssystems hinsichtlich Rechenleistung, Arbeitsspeicherverbrauch und Energiebedarf und die Verwundbarkeit des Systems gegenüber Schadsoftware.

Nach den bisherigen Ausführungen erscheint es so, als würde in jedem Fall nur die zu entwicklende Applikation an das Endgerät angepasst. In bestimmten Fällen ist es jedoch umgekehrt und man muss sich frühzeitig im Entwicklungsverlauf auf ein bestimmtes Gerät oder eine Gerätegruppe festzulegen, welches die gestellten Anforderungen am besten erfüllt ([16] S. 133).

3.1.2 Anforderungen durch Nutzerverhalten

Neben den Einschränkungen durch Hardware und Software hat auch das Nutzerverhalten der mobilen Nutzer maßgeblich Einfluss auf die Anwendungsentwicklung. Im Folgenden soll das Nutzerverhalten bei Benutzung eines mobilen Endgerätes dem der Benutzung eines stationären Rechners gegenüber gestellt werden.

Bei den meisten mobilen Endgeräten handelt es sich um Geräte, die den ganzen Tag mit sich geführt werden und zwischendurch immer kurz zum Einsatz kommen. Anders als beim Desktoprechner, wo man sich gezielt und länger in einer Anwendung aufhält, werden bei einem mobilen Endgerät verschiedenste Anwendungen meist nur kurz aufgerufen z.B. beim Eintragen eines Termins in den elektronischen Kalender.

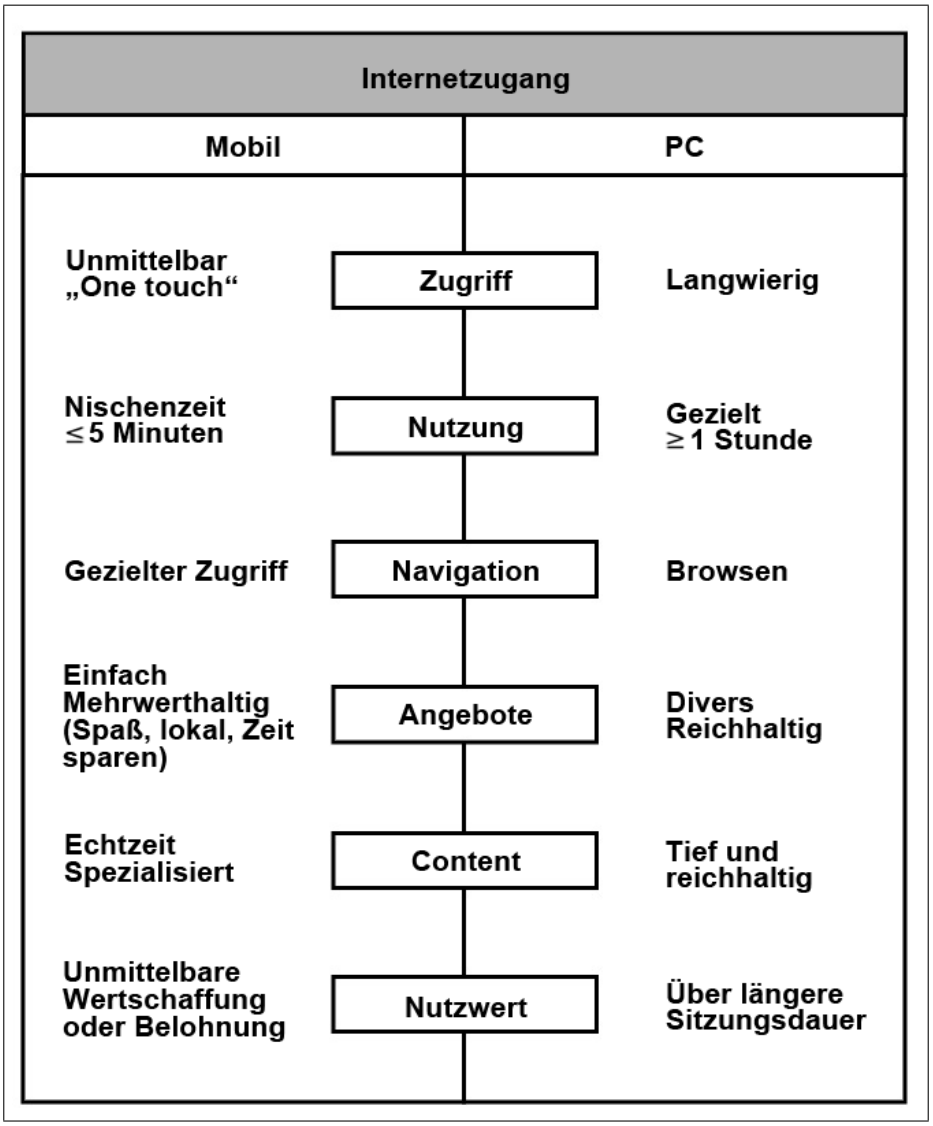

Abbildung 3.1: Nutzung des Internets Mobile / PC Quelle: [2] S. 116f

Untersuchungen hinsichtlich des Nutzungsverhaltens haben sich bisher vorwiegend mit der unterschiedlichen Benutzung des Internets befasst. Die Ergebnissen einer solchen Studie von Jörg Zobel ([2]) sind in Abbildung 3.1 zusammengefasst. Anders als am PC wird das Internet spontan über einen kurzen Zeitraum benutzt und man erhält einen Mehrwert über den schnellen Zugriff auf gezielte Informationen (vgl. auch [13] S. 241f).

Aus dem Nutzerverhalten ergeben sich zusammengefasst unter anderem folgende Anforderungen an mobile Software (vgl. [15], [17] S. 8f und [13] S. 243f):

- Schneller Anwendungsstart
- Feedback auf Benutzereingaben
- Simple Bedienung
- Schnelles Erreichen des Benutzungsziels

Wird die mobile Anwendung gestartet, erwartet der Nutzer bereits in kürzester Zeit eine lauffähige Anwendung. Die CPU Rechenleistung ist ein Faktor, der maßgeblich die Ladezeit einer Anwendung beeinflussen kann. Die mobile Anwendung muss also auf die Gesamtleistung des oder der Zielgeräte mit einem bestimmten mobilen Betriebssystems abgestimmt sein, um so eine schnelle Ladezeit zu garantieren.

Der Benutzer mobiler Anwendungen erwartet in jedem Fall ein schnelles Feedback auf seine Eingaben. Bei nur kurzen Wartezeiten sollte nach Möglichkeit ein Statusindikator angezeigt werden. Dieser verhindert, dass der Nutzer das Gefühl eines Systemabsturzes bekommt.

Eine Benutzung der Anwendung soll intuitiv ohne Hinzunahme eine Bedienungsanleitung erfolgen können. Um dies zu erreichen, ist insbesondere die Verwendungen von Standardelementen bei der Gestaltungen der Benutzeroberfläche ratsam. Hierdurch hat der Nutzer durch Konsistenz hinsichtlich des „Look and Feel“ gegenüber anderen Anwendungen einen Wiedererkennungseffekt und bekommt leichten Zugang zur Bedienung der Anwendung.

Außerdem muss dem Benutzer ermöglicht werden, innerhalb von kürzester Zeit sein Benutzungsziel also z.B. das Erlangen einer Information zu erreichen. Möglichkeiten dies zu erreichen, sind eine flache Navigationshierachie und vor allen Dingen die Anpassung des Funktionsumfangs. Soll von einer komplexen bereits vorhandenen Desktopanwen-

dung eine mobile Version entwickelt werden, kann man davon ausgegangen werden, dass nur ein Bruchteil der angebotenen Funktionen auch auf dem mobilen Endgerät sinnvoll genutzt werden kann (vgl. [15] S. 25). Apple geht davon aus, dass von ca. 80 Prozent der Benutzer nur ein eingeschränkter Funktionsumfang genutzt wird. Diesen gilt es individuell für die Anwendung zu ermitteln und die mobile Anwendung darauf abzustimmen. Des Weiteren müssen Art der Benutzereingabe und Benutzeroberfläche unter ergonomischen Aspekten angepasst werden (vgl. [15] S. 37). Die Benutzereingabe kann hier in vielfacher Hinsicht optimiert werden. Im Allgemeinen kann es durch die Reduzierung von Interaktionselement hinsichtlich ihrer Anzahl, dem Nutzer leichter gemacht werden zu Entscheiden was zu tun ist. Außerdem sollte der Eingabeaufwand des Benutzers dadurch reduziert werden, dass bevorzugt eine Auswahl getroffen werden kann anstelle einer eignen Eingabe. Hierzu bietet sich ein Pendant zum traditionellen „Drop Down“ oder auch scrollbare Tabellen an. Buttons oder andere Klickelemente sollten einen an das Gerät angepassten „Klickbereich“ haben, um Fehleingaben zu vermeiden. Für den iPod touch gibt Apple eine Größe von 44 x 44 Pixel als „sichere“ Größe an. Die Informationsdarstellung muss an die Auflösung des jeweiligen Endgerätes angepasst sein. Es muss ein Optimung zwischen dargestellerter Informationsmenge und Aufnahmevermögen des Nutzers erfolgen. Wichtige Informationen sind im oberen Bereich der Anwendung anzuordnen, damit sie sofort wahrgenommen werden.

Aus den hier genannten Punkten wird deutlich, dass Software für mobile Endgeräte stark an ein konkretes Gerät oder zumindest eine Kategorie von mobilen Geräten und deren Einschränkungen anzupassen sind. Dabei müssen im Falle von Android zum Beispiel Geräte mit verschiedensten Hardwarekonfigurationen als letztendliche Zielplattform in Erwägung gezogen werden. Bei Apple sieht es zurzeit so aus, dass iPod touch und iPhone in den verschiedenen Versionen zumindest gleiche Bildschirme und Eingabemöglichkeiten liefern. Ändern wird sich dies teilweise mit Einführung des iPad, welches zumindest einen größeren Bildschirm mit höherer Bildschirmauflösungen haben wird. Bisherige Anwendungen werden jedoch laut Apple (siehe [18]) problemlos auf dem iPad laufen und entweder in Originalgröße in Mitte des Bildschirms oder aber hochgerechnet auf dem ganzen Bildschirm dargestellt.

3.2 Funktionsumfang der mobilen LASAD Anwendung

Unter Berücksichtigung der Besonderheiten der Softwareentwicklung für mobile Endgeräte (vgl. 3.1) ergeben sich auch Konsequenzen für den Funktionsumfang. Bei komplexen Anwendungen wird mit hoher Wahrscheinlichkeit nur ein Bruchteil der Funktionen von allen Benutzern genutzt. Es ist also zu prüfen, welche Funktionen für die mobile Anwendung relevant sind und wie diese sich an die Anforderungen mobiler Endgeräte anpassen lassen.

Die derzeitige LASAD Google Web Toolkit (GWT)) Anwendung stellt dem Nutzer eine Vielzahl von Funktionen bereit. Basisfunktionen wie die Zugangskontrolle zum System (Login) und die Auswahl der zu bearbeitenden Map müssen in jedem Fall auch in der mobilen Anwendung vorhanden sein. Eine Map repräsentiert einen Datensatz auf dem mit den Elementen der Ontologie gearbeitet wird. Durch die Nutzerauthentifizierung ist es nach Betreten einer Map möglich, den anderen Nutzern gute Awareness Informationen darüber zu geben, wer auch an der Map arbeitet. Und für den nach 3.1.2 geforderten schnellen Zugriff auf Informationen ist es nötig, den gezielten Zugriff auf eine Map zu ermöglichen.

Kritischer zu betrachten sind die Funktionen aus dem Bereich der Map-Betrachtung und -Bearbeitung. Zweifellos müssen die Daten der Map in irgendeiner Form dargestellt werden. Ob und welche Funktionen zum Bearbeiten der Elemente portiert werden sollen ist genauer zu prüfen. Die Elemente werden im Folgenden als Box bezeichnet.

Zusammengefasst stehen dem Nutzer in der aktuellen Betaversion die in Tabelle 3.1 aufgelisteten Funktionen zur Bearbeitung der Daten zur Verfügung.

Die Funktionen 1–3 sollten als zusammenhängend betrachtet werden. Position und Größe einer Box sind maßgeblich entscheidend für die Einordnung der Box in den Gesamtkontext gegebenenfalls schon vorhandener Maps. Es erscheint zunächst sinnvoll, dem mobilen Nutzer die Möglichkeit zu geben, neue Boxen zu einer Map hinzuzufügen. Hierzu ist es nötig, dass der Nutzer die Inhalte einer Box eingeben kann, was im Vergleich zu Desktop durch die beschränkten Eingabemöglichkeiten erschwert ist (vgl. 3.1.1). Des Weiteren muss er die Box im Kontext der Map positionieren und ihre Größe möglicherweise anpassen. Hierzu ist es nötig, die bisherige Karte so darzustellen, dass der Nutzer zum Positionieren einen guten Überblick hat. Bei einer größeren Map könnte

Nr.	Beschreibung
1	Box erstellen
2	Box Position auf Karte verändern
3	Box Größe verändern
4	Inhalt in Box einfügen / bearbeiten
5	Boxen über Relationen miteinander verbinden
6	Relationen bearbeiten
7	Boxen und Relationen löschen
8	Bezug zwischen Transcript und Box herstellen

Tabelle 3.1: LASAD Funktionalität bei der Bearbeitung von Map Elementen

dies bei einigen Implementierungen zu einem Konflikt zwischen Übersichtlichkeit und geforderter Mindestgröße von „Klickelementen“ führen.

Die Inhalte einer Box zu bearbeiten (Funktion 4) ist technisch möglich und vom Nutzer erwünscht. Kritikpunkte sind auch hier wieder die beschränkten Eingabemöglichkeiten am mobilen Endgerät.

Um sinnvoll Relationen zu bearbeiten (Funktionen 5–7), bedarf es im Regefall eines guten Überblicks über die Map. Ob man diesen bei größeren Maps mit vielen Elementen auf einem mobilen Endgerät geben kann, ist bei jetzigen Hardwarerestriktionen durch niedrige Bildschirmauflösungen fraglich.

Einen Bezug zwischen Transcript und Box herzustellen (Funktion 8) ist auch bei eingeschränkter Übersicht möglich. Es ist jedoch zu überprüfen, ob es sich hierbei um eine wichtige Kernfunktionalität handelt.

Will man die Auswahl der Bearbeitungsfunktionalität abschließend klären, müssen in jedem Fall Benutzer der LASAD Software über ihr Nutzungsverhalten befragt werden. Studien mit der Betaversion werden aber erst im kommenden Monat beginnen, so dass es zur Zeit noch keine aktiven Nutzer gibt, die befragt werden könnten.

Eine erste Portierung auf ein mobiles Endgerät hat sich auf die Kernfunktionalität hinsichtlich der Betrachtung der am PC erstellten Informationen zu konzentrieren. Gleichzeitig sollten Erweiterungsmöglichkeiten zur Implementierung der Bearbeitungsfunktionalität berücksichtigt werden.

3.3 Benutzeroberfläche der mobilen LASAD Anwendung

In folgenden Abschnitt soll die angepasste Benutzeroberfläche einer mobilen argumentationsunterstüzenden Anwendung am Beispiel von LASAD diskutiert werden. Hierbei wird im ersten Teil 3.3.1 losgelöst von einer konkreten Implementierung auf dem iPod touch ermittelt welche Ein-/Ausgabemasken relevant sind. Anschließend wird in 3.3.2 untersucht, ob eine mobile GWT Permutation für die Implementierung auf dem iPod als mögliche Lösung in Frage kommt.

3.3.1 Allgemeiner Entwurf

In Abschnitt 3.1 wurde bereits festgestellt, wie Hard- und Softwarerestriktionen aber auch nicht zuletzt das Nutzungsverhalten des Anwenders großen Einfluss auf die Gestaltung einer mobilen Anwendungen haben. Es erscheint daher schon ohne größere Betrachtung als unwahrscheinlich, dass man eine Benutzeroberfläche einer Desktopanwendung 1:1 auf ein mobiles Endgerät übertragen kann.

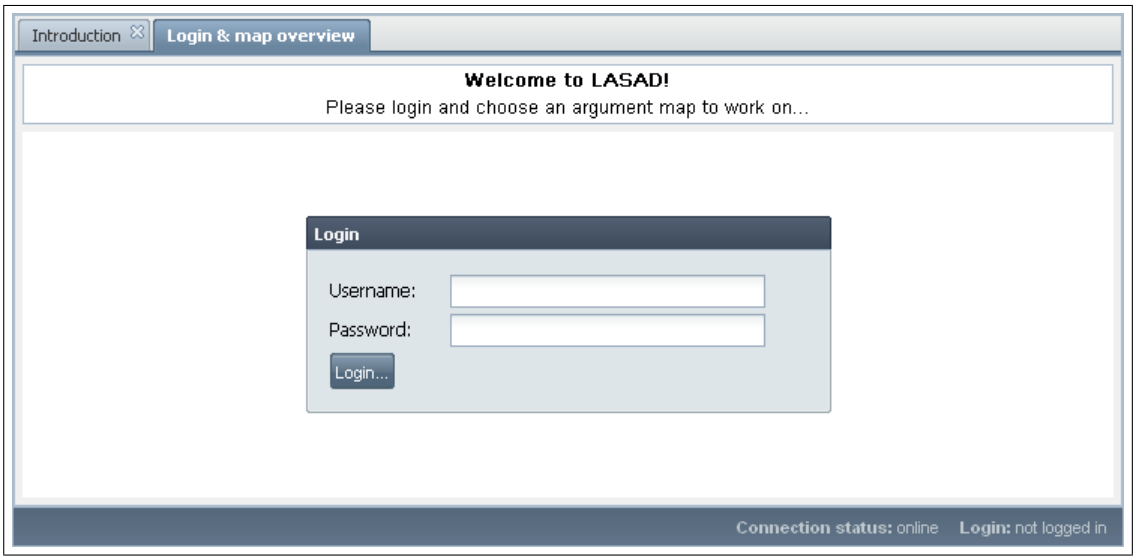

Abbildung 3.2: Loginmaske des LASAD GWT Clients

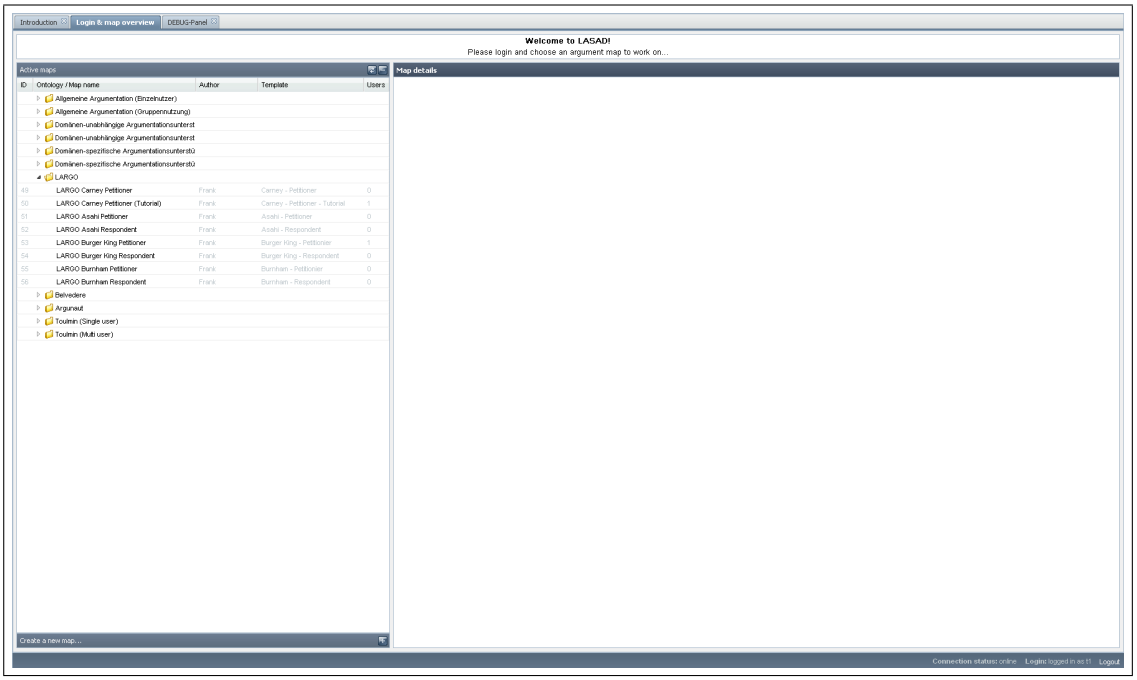

Abbildung 3.3: Übersicht aller Maps im LASAD GWT Client

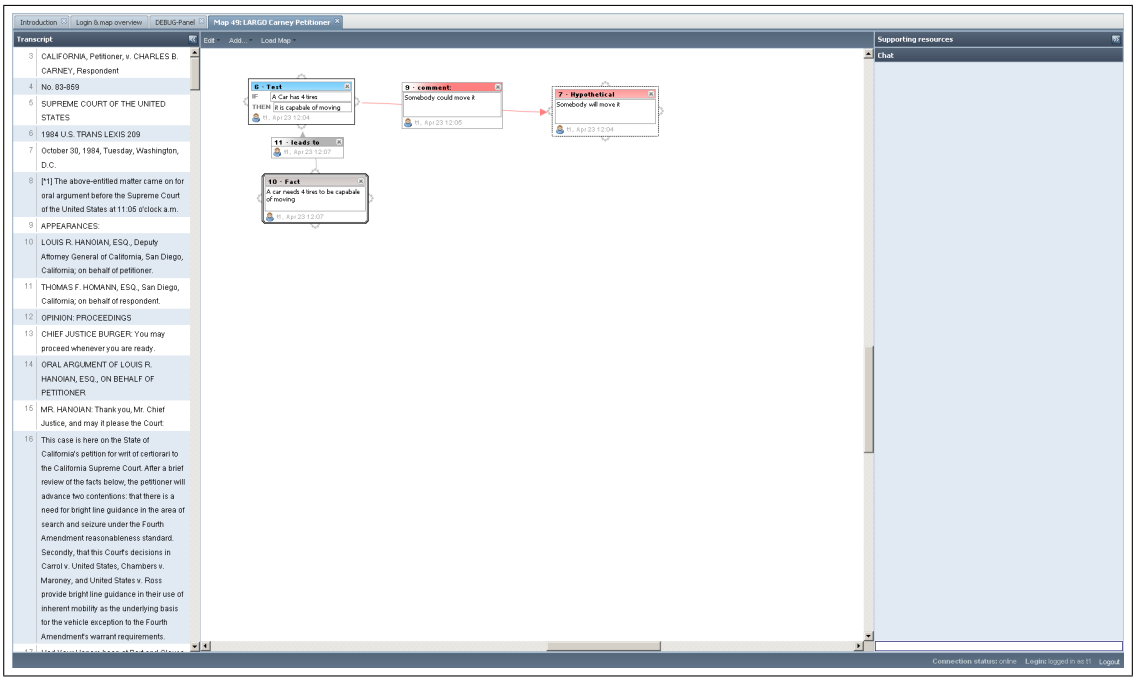

Abbildung 3.4: Diagramm Ansicht des LASAD GWT Clients

Der bisherige Desktop LASAD Client ist mittels GWT realisiert. Aus geschrieben Java Code wird unter anderem eine Benutzeroberfläche aus HyperText Markup Language (HTML) und Cascading Style Sheets (CSS) generiert, die weitere Ressourcen wie Bilder zur Darstellung einbinden. Dargestellt wird diese Benutzeroberfläche dann mit Hilfe eines Webbrowsers.

Die Benutzeroberfläche der LASAD Betaversion gliedert sich im Groben in 3 Masken. Bei Start der Anwendung wird zunächst die Login Maske (siehe Abbildung 3.2) angezeigt, nach erfolgreichem Login wechselt die Ansicht in die Lobby (siehe Abbildung 3.3). Hier werden alle vorhanden Maps aufgelistet und es besteht die Möglichkeit einer Map zu beizutreten. Durch das Betreten einer Map wird in die „Diagramm Ansicht“ (siehe Abbildung 3.4) gewechselt. In der Mitte dieser Ansicht wird in Form eines Diagramms die Map mit ihren Elementen dargestellt. Auf der linken Seite, in einem wahlweise einklappbaren Teilfenster wird ein Transcript eines Textes eingeblendet, der Grundlage für eine Argumentation bilden kann. Auf der rechten Seite wird im „Tutorial Modus“ ein Einführungstutorial angezeigt, durch das man sich durcharbeiten kann.

Die 1:1 Umsetzung der Diagramm-Ansicht mit ihren Teilfenstern auf einem mobilen Endgerät ist wegen der Hardwarerestriktionen offensichtlich im Allgemeinen nicht möglich. Bei einer Bildschirmauflösung von 480x320 Pixel beim iPod touch ist es nicht möglich, in vertretbarer Schriftgröße derart viele Informationen darzustellen. Untermauert wird dies in Abbildung 3.6 und Abbildung 3.5 mit unangepasstem GWT-Client, welcher im nächsten Abschnitt 3.3.2 noch genauer diskutiert werden. Folglich muss die GWT-Benutzeroberfläche in mehrere Masken unterteilt werden, um alle Daten sinnvoll anzeigen zu können.

Eine mögliche Aufteilung ist die in Tabelle 3.2 aufgeführte. Sie ist eine von vielen Möglichkeiten, alle nötigen Informationen unter Berücksichtigung der Hardwareeinschränkungen anzuzeigen. Inwieweit einzelne Masken bei der Darstellung wiederum kombiniert oder verknüpft werden, hängt von einer individuellen mobilen Plattform (Kombination aus Hardware und Betriebssystem) ab. Für den iPod touch wird eine Lösung in Abschnitt 4.2 dargestellt.

Nr.	Maske	Beschreibung
1	Startbildschirm	Anzeige einer Loginmaske
2	Übersicht	Auflistung aller vorhandenen Projekte nach Ontologie
3	Projektdetailseite	Anzeige allgemeiner Projektdaten
3.1	Transcript Anzeige	Tabellarische Auflistung der einzelnen Transcript Zeilen. Jede Zeile ist klickbar und zeigt Verlinkungen zu Boxen
3.2	Box Auflistung	Tabellarische Auflistung aller vorhandenen Boxen nach Typ. Boxen sind klickbar für weitere Details
3.3	Diagramm	Vereinfachte grafische Darstellung der Daten als Diagramm
4	Box Detailseite	Zeigt die Eigenschaften einer konkreten Box an und bietet Möglichkeit, sich die Verlinkungen zu anderen Boxen und dem Transcript anzuzeigen
5	Transcript Zeile Detailseite	Zeigt kompletten Text einer Transcript-Zeile an und bietet die Möglichkeit, sich Verlinkungen mit Boxen anzeigen zu lassen
6	Verlinkungen von Box ausgehend	Zeigt mit welchen Boxen oder Textabschnitten eine Box in Verbindung steht
7	Verlinkungen von Transcript Zeile	Zeigt ob eine Transcript-Zeile in Verbindung mit ein oder mehreren Boxen steht

Tabelle 3.2: Einzelne Masken der mobilen Benutzeroberfläche

3.3.2 GWT im mobile Safari

Der vorhandene LASAD Client ist mit Hilfe von GWT realisiert. In compilierter Form ist die Benutzeroberfläche theoretisch in jedem Browser lauffähig. Auf Grund der stark browserspezifischen Umsetzung von JavaScript und CSS kann es jedoch nötig sein, für jeden Browser eine individuelle Anpassung eine so genannte Permutation zu erstellen. Bezogen auf den auf dem iPod touch installierten mobile Safari ist also eine angepasste Permutation möglich. Ob hierdurch ausreichend auf die Besonderheiten des mobilen Endgerätes eingegangen werden kann und ob ein GWT Client überhaupt für den Einsatz auf einem mobilen Endgerät geeignet ist, soll im Folgenden diskutiert werden.

Auf dem iPod touch ist werksseitig bereits mobile Safari als Webbrowser installiert. Wird nun über die Adresseingabe im Browser der GWT Client aufgerufen so stellt sich, je nach Ausrichtung des iPod der GWT Client wie in Abbildung 3.5 oder Abbildung 3.6 dar. Im Landscape Modus (bei Ausrichtung des iPods mit langer Kante nach oben Abbildung 3.5) werden alle Bedienelemente des Clients in verkleinerter Form dargestellt. Auf Grund der geringen Größe sind Bedienelemente jedoch in dieser Größe nur bedingt benutzbar. Bei anderer Ausrichtung (kurze Kante nach oben Abbildung 3.6), dem so genannten Portrait-Modus wird der Client auf der rechten Seite abgeschnitten und es muss bereits jetzt gescrollt werden, um alle Bedienelemente zu sehen. Zudem ist aufgrund der niedrigen Auflösung ohne Zoomen kaum der Text zu erkennen und die kleinen Bedienelemente sind nicht zielgerichtet klickbar.

Eine Notlösung bietet die Zoomfunktion. Wird die Benutzeroberfläche stark vergrößert (Abbildung 3.7), so sind die Buttons zielgerichtet klickbar. Ein großer Nachteil für den Nutzer ergibt sich jedoch dadurch, dass er so den Überblick über die Gesamtanwendung verliert und das Zoomen in mehreren Zwischenschritten mitunter sehr zeitaufwendig ist. Des Weiteren ist es in diesem Modus trotzdem nicht möglich, Boxen zu verschieben, miteinander zu verbinden oder den Bezug zwischen Text und Box herzustellen. Die von der Desktopanwendung übernommene Eingabe „Klicken - Gedrückthalten - Ziehen“ des Nutzers wird vom iPod als Geste erkannt und im Browser als „Verschiebe den Bildschirmausschnitt“ interpretiert

Viele dieser Anzeigeprobleme lassen sich mit Sicherheit durch eine angepasste Permutation lösen, jedoch ist zu Begutachten, in wie weit sich GWT generell für den Einsatz auf einem mobilen Endgerät eignet. Wird ein GWT Client im Browser aufgerufen, so

Abbildung 3.5: LASAD GWT Client in mobile Safari - Landscape Ansicht

werden über HyperText Transfer Protocol (HTTP) HTML Code, JavaScript Code, CSS Code und Bilder übertragen. Bei Abschluss der Übertragung wird dann der HTML und CSS Code entsprechend angezeigt und die Skripte ausgeführt. Diese technische Lösung hat im Kontext eines mobilen Endgerätes mehrere Nachteile. Mobile Endgeräte sind im Normalfall nicht per Kabel an ein lokales Netzwerk oder das Internet angebunden. Die Datenverbindung erfolgt drahtlos über WLAN oder eine Mobilfunktechnologie. Auch wenn die Übertragungsgeschwindigkeiten immer weiter verbessert werden, so ist Traffic begrenzt und oder verursacht hohe Kosten. Bei der Nutzung eines GWT Clients entsteht neben der Übertragung der Inhalts relevanter Daten ein nicht unerheblichen Overhead in Form von HTML, Javascript und CSS sowie den Bildern von Schaltflächen für die Benutzeroberfläche. Neben dem Übertragungsvolumen kommt eine hohe Ladezeit hinzu. Wie bereits in 3.1.2 beschrieben, erwartet der Nutzer eines mobilen Endgeräts eine betriebsfähige Applikation bereits nach wenigen Sekunden, was also möglicherweise schon bei der GWT Applikationsübertragung nicht erreicht werden kann.

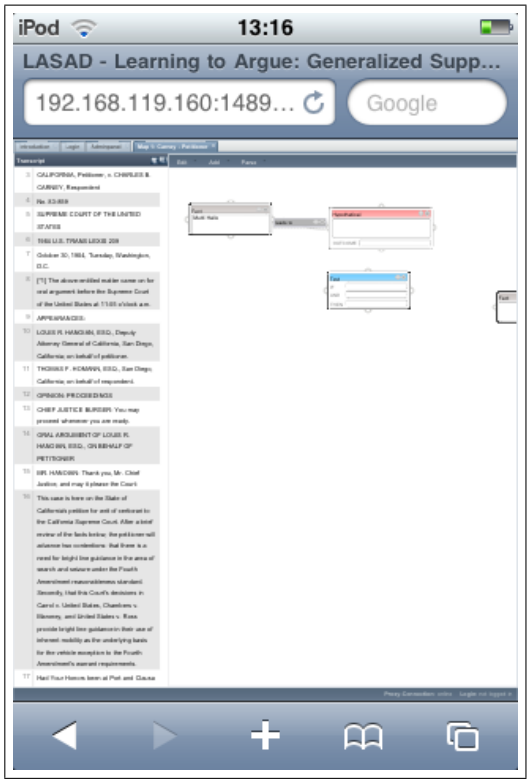

Abbildung 3.6: LASAD GWT Client in mobile Safari - Portrait Ansicht

Zusammengefasst ergeben sich insbesondere folgende Probleme beim Einsatz eines GWT Clients im mobilen Kontext:

- Darstellungsprobleme auf Grund der geringen Bildschirmauflösung
- „Klicken - Gedrückthalten - Ziehen“ wird nicht als solches interpretiert
- GWT Traffic Overhead
- GWT Anwendungsladezeit

Schreibt man eine angepasste GWT Permutation für den Safari Mobile, kann man einzig die Darstellungsprobleme beheben oder zumindest vermindern. Alle übrigen Probleme lassen sich durch eine eigene Permutation jedoch nicht beheben. Gerade das Fehlen einer Maus als Eingabegerät macht es schwierig, alle möglichen Nutzereingaben der Desktopversion auf eine mobile GWT Version zu übertragen. Zusammengefasst ist mittels GWT zumindest keine optimale Realisierung einer mobilen LASAD Anwendung

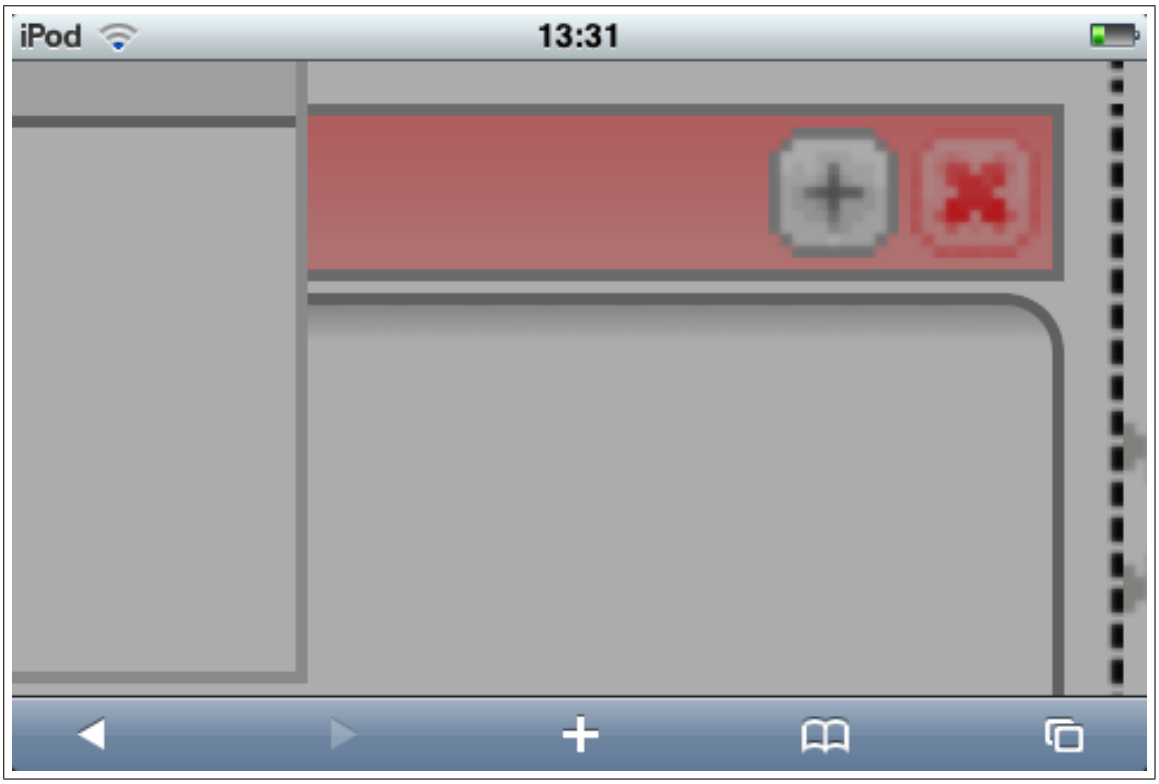

Abbildung 3.7: LASAD GWT Client in mobile Safari - Zoom auf Bedienelemente

möglich.

Im Bezug auf den iPod touch bietet sich eine eigenständige und native „App“ als Lösung für die Umsetzung der mobilen LASAD Anwendung an. Das iPhone SDK bringt hierzu bereits Standardfunktionalität und Benutzeroberflächenelemente mit (weiterführend in 4.1). Diese Komponenten sind bereits im Einsatz bewährt und dem Nutzer durch die von Apple mitgelieferte Standardsoftware bekannt. Hierdurch tritt beim Nutzer ein Wiedererkennungseffekt hinsichtlich des „Look and Feel“ ein, welcher den schnellen Einstieg in die Anwendung ermöglicht. Zudem wird ein unnötiger Trafficoverhead vermieden, da nur noch inhaltlich relevante Daten übertragen werden und nicht die Anwendung selbst. Bei optimierter Programmierung dürfte zudem eine angemessene geringe Ladezeit der Applikation garantiert werden.

Action	Kurzbeschreibung
LOGIN	Übersendung der Logindaten zur Authentifizierung
LIST	Anfrage nach allen vorhanden Maps
JOIN	Beitritt zu einer Map, in Antwort aktueller Datenstand
LEAVE	Verlassen einer Map

Tabelle 3.3: Über doActionPackage ausführbare „Actions“

3.4 Datenaustausch mit dem Server

Die LASAD Architektur (siehe Abbildung 2.1) bietet grundsätzlich 2 Schnittstellen für Clients an. Zum einen ist im Rahmen der GWT Nutzung der Zugriff über das RemoteServiceServlet und allgemein über einen Webservice möglich. Für die mobile LASAD Anwendung eignet sich insbesondere der Zugriff über den WebService. Als Kommunikationsprotokoll benutzt der Webservice SOAP, über welches nach außen 2 Methoden bereitgestellt werden:

- doActionPackage
- doPolling

Mittels doActionPackage können gezielte Anfragen an den Server gerichtet werden, mit doPolling bekommt man vom Server Änderungen mitgeteilt. Durch den Aufruf von doActionPackage auf dem Server können so genannte Actions ausgeführt werden. Ausführbare Actions sind unter anderem die vier in Tabelle 3.3 aufgeführten.

Darüber hinaus sind noch weitere Actions insbesondere hinsichtlich der Bearbeitung von Mapdaten verfügbar. Für den in Abschnitt 3.2 festgelegten Funktionsumfang, der im Wesentlich dem eines Viewers entspricht, sind die aufgeführten Actions bereits ausreichend. Im Rahmen der Implementierung muss also der Aufruf der doActionPackage Methode des WebServices mittels SOAP realisiert werden, um die benötigten Daten vom Server zu bekommen.

4 Implementierung

4.1 Implementieren von iPod touch Anwendungen

Die Entwicklung von iPod touch Anwendungen geschieht im Allgemeinen mit Hilfe des von Apple veröffentlichten iPhone Software Development Kit (SDKs). Wie in Abschnitt 2.2.3 bereits erwähnt, läuft mit iPhone OS auf dem iPod touch das selbe Betriebssystem, wie auf dem iPhone und dem iPad.

Über die Apple Developer Seiten ([19]) kann man die aktuelle Version des SDK nach Registration kostenlos herunterladen und benutzen. Eine Einschränkung hierbei ist, dass das SDK nur für Mac OS verfügbar ist. Im Umfang des SDKs befinden sich eine Vielzahl von Teilanwendungen. Als Editor fungiert XCode, mit dem auch für Mac OS entwickelt wird. Mit Hilfe des mitgelieferten Interface Builder lassen sich Elemente für die Benutzeroberfläche zusammenstellen und parametrisieren. Des Weiteren sind ein Compiler für Objective-C Code, ein Simulator für iPhone OS und eine Vielzahl kleiner Hilfsprogramme enthalten.

Die Standardprogrammiersprache für iPhone OS Anwendungen ist Objective-C. Entwickelt wird im allgemeinen nach dem Model-View-Controller Pattern. Erwähnenswert ist die Tatsache, dass bisher iPhone OS keine Multitaskingfähigkeit gegenüber Fremdsoftware bereitstellt. Im Allgemeinen kann also nur eine Applikation gleichzeitig ausgeführt werden, welche beim Wechsel zu einer anderen beendet wird. Für iPhone OS 4.0 wurde die allgemeine Multitaskingfähigkeit als neues Feature angekündigt (siehe [12]). Über den Apple iPhone Developerbereich erhält man zahlreiche Handbücher, Guidelines, Tutorials sowie Beispielanwendungen. Plant man eine Anwendung für den App-Store zu entwickeln, so sollte man frühzeitig die „Human Interface Guidelines“ von Apple ([15]) lesen. In ihr werden genaue Vorgaben bezüglich der Benutzeroberfläche gemacht. Soll eine Applikation später über den App-Store verkauft werden, so müssen diese

Vorgaben eingehalten werden.

Zu Beginn der Entwicklung erstellt man über XCode, ähnlich wie bei anderen integrierte Entwicklungsumgebungen (IDEs)), ein Projekt. Für iPhone OS Anwendungen sind folgende Projekttemplates verfügbar:

- Navigation-based Application
- OpenGL ES Application
- Tab Bar Application
- Utility Application
- View-based Application
- Window-based Application

Eine „Navigation-based" Application bietet, mit Hilfe einer so genannten Navigation-Bar, die Möglickeit in der Anwendung vor und zurück zu navigieren. Mittels OpenGL können auch grafisch aufwendigere Applikationen realisiert werden, insbesondere also Spiele. Die Tab Bar ist das Pendant zu „Reitern" in anderen Programmiersprachen. Durch das klicken eines Tab Bar Buttons wechselt man zwischen verschiedenen Masken hin und her. Eine Utility Application ist nach Definition in der „Human Interface Guide" eine kleine Anwendung, bei der wenig User-Eingaben erforderlich sind und bei der nur eine begrenzte Anzahl von Informationen angezeigt werden. Die standard Wetter-App von Apple ist ein gutes Beispiel für eine Utility App.

Diese ersten vier Templates kommen schon mit viel Standardfunktionalitäten und ermöglichen eine schnelle Entwicklung der speziellen Anwendungstypen. Letztere sind im Vergleich dazu minimale Templates, die dem Entwickler nicht viel Arbeit abnehmen, aber daher auch mehr Freiheit bei der Entwicklung lassen.

Beim Test einer Applikation im iPhone Simulator ist zu bedenken, dass sich dieser teilweise anders verhält, als das tatsächliche Endgerät. Mitunter kann der Simulator auf mehr Libraries zurückgreifen, als es das Endgerät kann. Zudem berücksichtig er die Hardwarebeschränkungen des tatsächlichen Gerätes nur eingeschränkt. Ein Beispiel dafür ist das Verhalten des Auto Release Pools, der für die automatische Freigabe von Variablen zuständig ist. Bedingt durch mehr zur Verfügung stehenden Arbeitsspeicher, gibt der „Auto Release Pool" oftmals Speicher erst vergleichsweise spät wieder frei.

Fehler im Release Management werden hierdurch im Simulator möglicherweise übersehen. Es ist also unbedingt nötig, in kurzen Abständen auch auf dem Endgerät die Applikation hinsichtlich ihrer Funktionsfähigkeit zu untersuchen.

Vor dem ersten Testlauf auf einem iPod touch Testgerät sind einige Hürden zu nehmen. Zunächst benötigt ein Entwickler ein von Apple ausgegebenes Entwickler-Zertifikat, mit dem später die eigenen Anwendungen signiert werden. Er benötigt eine allgemeine oder Uni-Entwicklerlizenz und kann sich dann über den Entwicklerbereich das Zertifikat generieren und lokal installieren. Anschließend muss die Geräte-ID des Testgerätes online hinterlegt werden. Ebenso muss für jede Applikation online eine App–ID generiert werden. Unter Angabe von Certifikat , Gerät und App–ID kann dann ein so gennantes „Provisioning Profile“ erzeugt werden. Dieses muss herunter geladen und über XCode oder iTunes auf den iPod touch übertragen werden. Durch die Installation wird der iPod touch zu einem autorisierten Testgerät. Mittels „Provision Profile“ ist auf dem Gerät definiert welche Applikation, von welchen Developern ausgeführt werden dürfen. Nach einem bestimmten Zeitraum läuft das Provision Profile ab. Es muss dann erneut generiert und auf dem Testgerät installiert werden. Das übertragen der Applikation auf das Endgerät geschieht direkt über XCode. Das angeschlossene Testgerät kann dort als Ziel-SDK ausgewählt werden. Mit Klicken von „Build and Run“ wird die compelierte App auf das Gerät übertragen und ausgeführt.

4.2 Implementierung der Benutzeroberfläche

Nachdem in Abschnitt 3.3 losgelöst vom iPod ermittelt wurde, welche Masken die Benutzeroberfläche einer mobilen Anwendung umfassen muss, soll nun vorgestellt werden welche Elemente zur Umsetzung der iPod-Benutzeroberfläche zur Verfügung stehen und wie diese in einer Beispielanwendung kombiniert wurden.

Apple hat für iPhone Entwickler ein Dokument speziell in Hinblick auf die Gestaltung der Benutzeroberfläche von Apps herausgegeben, die „ Human Interface Guidelines “ ([15]). In diesem wird zu Beginn dringlich darauf hingewiesen, dass es in jedem Fall erforderlich ist, eine speziell auf die Plattformeigenschaften angepasste Benutzeroberfläche zu gestalten. Um diese zu entwerfen, sollen nach Möglichkeit, die von Apple bereitgestellten Standard-Benutzeroberflächen Elemente, verwendet werden. Sie wurden

speziell für das iPhone OS entworfen und die entsprechenden Ressourcen sind bereits im iPhone OS integriert. Dadurch, dass eine Vielzahl im Apple Store verkäuflichen Applikationen diese Standardelemente benutzen, besteht für den Nutzer ein hoher Wiedererkennungswert. So muss er sich nicht lange in eine Applikation einarbeiten, sondern kann auf Grund der eingesetzten Elemente auch ihre Funktion leicht erschließen. Gerade dieses Argument spricht für die Verwendung der iPhone OS Standardelemente bei der Umsetzung der Benutzeroberfläche. Zudem hält man sich so die Möglichkeit offen die Applikation später in den App-Store einzustellen, wo es für fast alle Anwendungen vorgeschrieben ist die Standard Elemente zu benutzen.

Im „Interface Builder“ (vgl. 4.1) stehen dem Entwickler diese Standardelemente zur Verfügung (siehe Abbildung 4.2). Die Auswahl an Elementen ist gering, jedoch lassen sich einige Elemente über ihre Parameter in Aussehen und Verhalten noch weiter anpassen. Zunächst soll nun ein Überblick über wichtige Standardelemente einer iPod Benutzeroberfläche gegeben werden.

Oberstes Element einer Anwendungsdarstellung ist ein so genanntes „Window“ Element. Jede Anwendung hat exakt **ein** Window und alle übrigen anzuzeigenden Elemente werden diesem untergeordnet. Dem Window ordnet man dann im Programmablauf einen einzelnen oder eine Abfolge von so genannten Views zu.

Ein View ist im Allgemeinen eine leere Fläche, auf der man dann weitere Elemente wie Buttons, Labels und Eingabefelder anordnen kann. Im Interface Builder stehen dem Entwickler bereits mehrere, für bestimmte Aufgaben spezialisierte, Views zur Verfügung. Basisklasse aller Views ist die Klasse UIView (vgl. Abbildung 4.1).

Eine Kategorie von spezialisierteren Views sind die Data Views (siehe Abbildung 4.2). Views aus der Kategorie Data View bieten die Möglichkeit bestimmte Arten von Informationen anzuzeigen und liefern bereits einige Funktionalität, um auf Benutzereingaben zu reagieren.

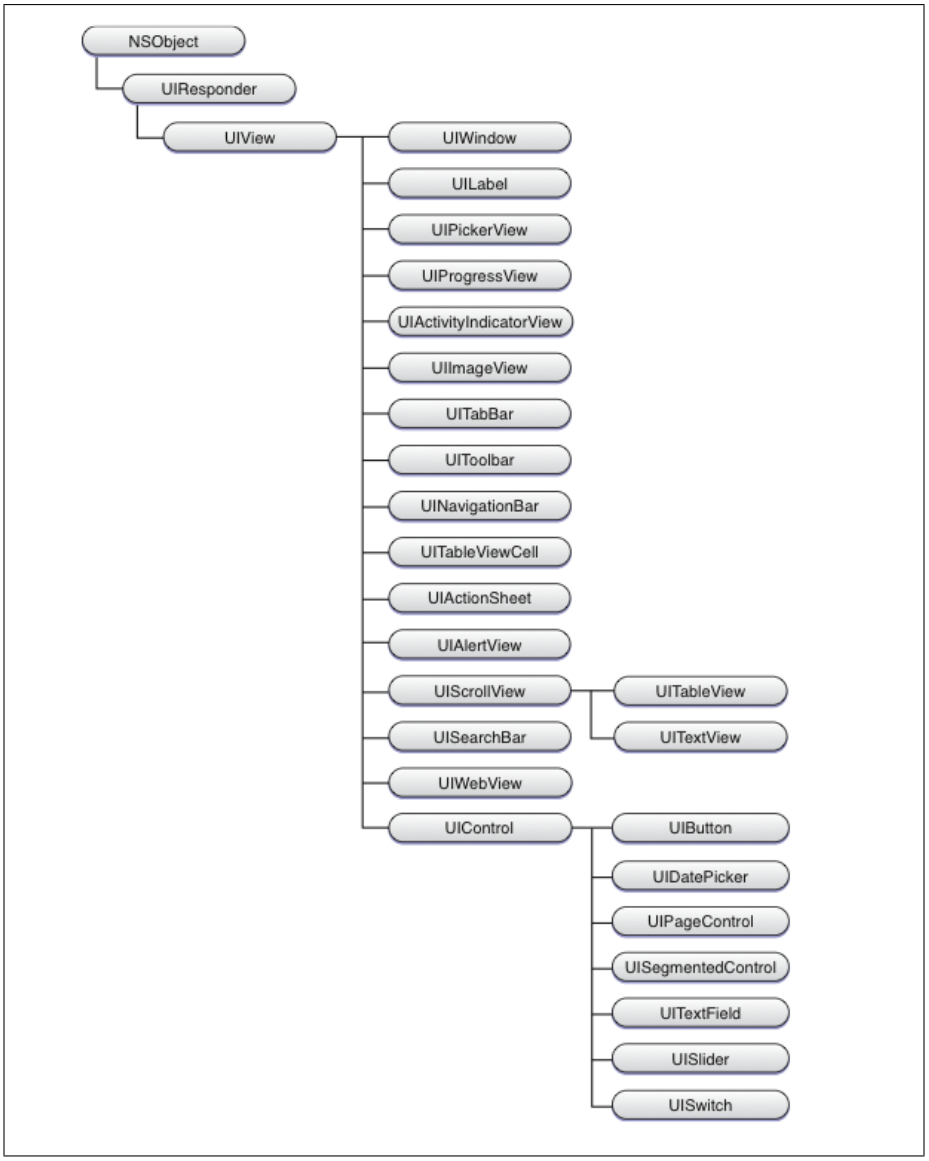

Abbildung 4.1: View Vererbungshierachie Quelle: [3]
S. 50

Abbildung 4.2: Interface Builder - Auswahl von Elementen aus der Library

Mittels eines „Table Views“, der „Table View Cells“ darstellt, lassen sich große Mengen, gleichartiger Informationen, in einer Tabelle übersichtlich darstellen. Funktionen wie scrollen innerhalb der Tabelle oder anklicken von einzelnen Tabellenzeilen werden bereits Frameworkseitig mitgeliefert. Es gibt die Möglichkeit verschiedene vordefinierte Table Cell Layouts zu benutzen oder eine komplett eigene Tabellenzelle zu definieren. Darüber hinaus können, über die Eigenschaften des Table Views, in verschiedener Weise die Auflistung der Zellen beeinflusst werden. So ist es zum Beispiel möglich, neben der einfachen aufeinander Folge von Zeilen, diese in Sektionen gruppieren zu lassen. Geht man die Data Views weiter in Reihenfolge von Abbildung 4.2 durch, so gibt es noch einen „Image View“ zur Anzeige eines Bildes, den „Web View“ zur Darstellung von Webinhalten und den „Map View“ um Kartenausschnitte von Goolge Maps anzuzeigen. Mittels eines „Text View“ kann man einen längeren Textabschnitt anzeigen lassen, der bei Bedarf dann auch über den Bildschirm nach unten hinaus geht und scrollbar ist. Neben der Darstellung kann man bei entsprechender Parametrierung auch das Editieren des angezeigten Textes ermöglichen. Eine sehr freie Darstellungsoption, da man hier verschiedene andere Darstellungsarten einbetten kann, bietet der „Scroll View“. Dieser dient dazu Inhalte darzustellen, die horizontal und vertikal über die Größe des Applikationsfenster hinausgehen. Die letzten beiden DataViews, der „Picker View“ und der „Date Picker“, entsprechen einem klassischen „Drop Down“ Feld einer Desktop Anwendung angepasst an die Anforderungen eines mobilen Endgerätes. Anstatt alle Auswahlmöglichkeiten gleichzeitig untereinander darzustellen, was bei mehreren Möglichkeiten schnell zur Überschreitung der Fensterhöhe führen würde, werden nur eine Anzahl von Auswahlmöglichkeiten, die sich vor oder nach der aktuell ausgewählten befinden, dargestellt. Durch eine Touch-Geste kann der Nutzer dann durch alle Möglichkeiten scrollen und die gewünschte auswählen. Der „Date Picker“ ist ein bereits mit Daten befüllter Picker View. Er bietet nach Parametrierung generierte Termine mit Uhrzeiten zur Auswahl an.

Aus allgemeinem Wissen der Softwareentwicklung kann man davon ausgehen, dass nur kleine Anwendungen mit einer Art von DataView auskommen. Im Normalfall dürfte eine Kombination oder Aufeinanderfolge von Data Views mit verschiedensten Nutzereingabemöglichkeiten zum Einsatz kommen. Durch gezielte Zuweisung von Views zum Window kann bestimmt werden, welcher View angezeigt wird und durch Nutzereingaben kann dann gegebenenfalls ein anderer View angezeigt werden.

Zur Realisierung einer zielgerichteten Anzeigereihenfolge, bei der der Nutzer leicht vor und zurück navigieren kann, bietet die Library einen Navigation Controller. Dieser verwaltet einen View Stack, so dass der Nutzer immer zurück navigieren kann. Betrachtet man andere iPod-Applikation, so wird der Navigation Controller oftmals in Kombination mit Table Views verwendet. Wird eine Tabellenzeile angeklickt, gelangt der Nutzer in eine tiefere Eben der Navigationshierachie. Üblicherweise wird ein Navigation Controller durch eine „Navigation Bar“ am oberen Bildschirmrand visualisiert. Sie beinhaltet den Titel des aktuellen Views und einen pfeilförmigen Button, über den zurück navigiert werden kann.

Möchte man verschiedene Views oder ganze View Hierarchien schnell parallel zugreifbar machen, kann man den so genannten „Tab Bar Controller“ verwenden. „Ein Tab Bar Controller“ verwaltet mehrere Tabs zwischen denen der Benutzer, mittels der üblicherweise am unteren Bildschirmrand befindlichen „Tab Bar Leiste“, hin und her wechseln kann. Vergleichbar ist dies mit den „Registerkarten“ in gängigen Desktopanwendungen.

Zur Realisierung der LASAD Applikation bietet sich eine Vielzahl von Kombinationsmöglichkeiten der zur Verfügung stehenden Elemente an. Die gewählte Lösung ist in Abbildung 4.3, Abbildung 4.4 und Abbildung 4.5 zu sehen. Alle in Abschnitt 3.3 ermittelten Masken wurden bei der Realisierung durch Standardelemente abgebildet.

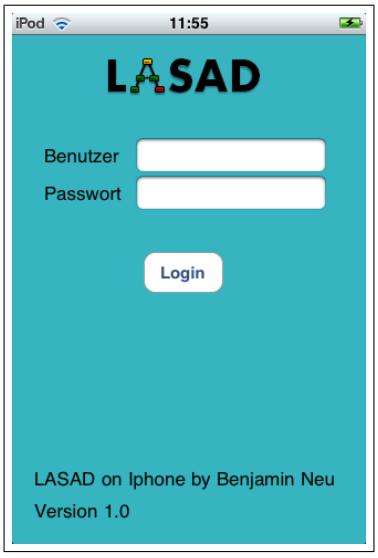

Abbildung 4.3: Ebene 1 - Startbildschirm

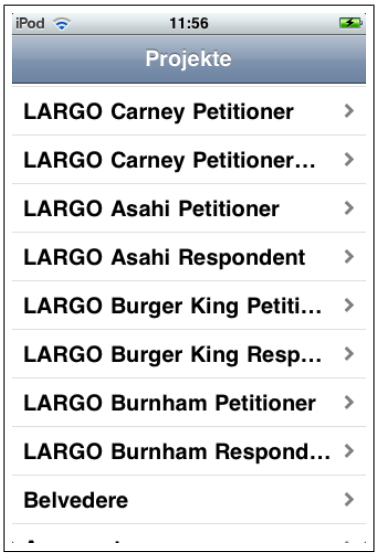

Abbildung 4.4: Ebene 2 - Liste aller Maps

Die Benutzeroberfläche gliedert sich in drei große Ebenen, hier repräsentiert durch die drei Abbildungen (Abbildung 4.3, Abbildung 4.4 und Abbildung 4.5). Abbildung 4.3 zeigt den geforderten Startbildschirm (Maske 1 nach Tabelle 3.2). Über das simple Eingabeformular kann sich der Benutzer Authentifizieren. Bei einem erfolgreichem Login gelangt er in die 2. Ebene (vgl. Abbildung 4.4). Mit Hilfe eines TableViews werden hier alle vorhandenen Projekte bzw. Maps aufgelistet (Maske 2). Über Scrollen und anschließendes Anklicken einer Zeile kann der Nutzer gezielt einer Map beitreten. Mit dem Betreten einer Map wechselt der Nutzer in die 3. Ebene der Benutzeroberfläche (vgl. Abbildung 4.5). Über einen Tab Bar Controller kann der Nutzer zwischen den Tabs „Home“, „Protocol“, „Boxes“ und „Diagram“ wechseln. Der „Home“ Tab entspricht hierbei der geforderten Maske 3. Es werden der Mapname und die auf der Map derzeit aktiven Benutzer angezeigt. Im „Protocol“ Tab werden zunächst tabellarisch verkürzt alle Zeilen des Transcripts abgetragen. Durch tieferes Navigieren in diesem Tab können die einzelne Zeilen des Transcripts komplett angeschaut und auch die Verlinkungen zu Boxen ermitteln werden (Maske 3.1, 5 und 7). Maske 3.2 wird im Tab „Boxes“ umgesetzt. Die Boxen der aktuellen Map werden, nach Typen gruppiert, mittels Table View aufgelistet. Tieferes navigieren ermöglicht die detaillierte Betrachtung der Boxen und ihrer Verlinkungen zu anderen Boxen oder zum Transcript (Masken 3.2, 4, 6 und 7). In dem letzten „Diagram“ Tab werden in Form eines stark vereinfachten Diagrams

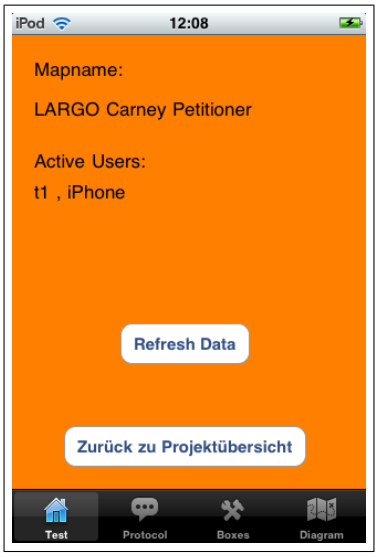

Abbildung 4.5: Ebene 3 - Projektübersicht „Home“

Boxen und ihre Verbindungen untereinander visualisiert (Maske 3.3). Boxen werden hierbei durch Buttons dargestellt, die dann über Linien miteinander verbunden sind. Die Anordnung der Boxen ist an die Positionierung im GWT Client, unter Berücksichtigung eines kleineren Koordinatensystems angelehnt.

4.3 Client-Server Kommunikation

In Abschnitt 3.4 wurde bereits festgelegt, dass eine Kommunikation zwischen Client und Server mittels SOAP geschehen soll. Trotz allgemein weiter Verbreitung dieser Kommunikationsart, bietet das iPhone OS in der aktuellen Version 3.1 keine eingebaute SOAP Unterstützung.

Eine SOAP Kommunikation kann grundsätzlich über eine Vielzahl von Protokollen geschehen. Für die Kommunikation mit dem LASAD Server soll HTTP als Transportprotokoll zum Einsatz kommen. Mittels HTTP-POST wird eine SOAP Nachricht an den Server geschickt und dieser antwortet mit einem HTTP-RESPONSE. Die zu übertragende Nachricht selbst besteht aus einem XML-Konstrukt. Technisch gesehen muss also die Möglichkeit vorhanden sein, einen String (XML Nachricht) über HTTP-POST an den Server zu senden und im Anschluss die HTTP-RESPONSE auszuwerten.

Diesen Anforderungen kann im Kontext von iPhone OS, unter anderem mit den Klassen NSMutableURLRequest und NSURLConnection, nachgekommen werden. Mit Hilfe dieser Klassen kann man einen individuellen HTTP-POST Aufruf senden und dessen Antwort später verarbeiten.

Die im LASAD Kontext an die Methode doActionPackage des WebService zu versendenden Soap-Nachrichten haben hierbei im Allgemeinen einen Aufbau wie in Abbildung 4.6.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ejb="http://ejb.lasad/">
   <soapenv:Header/>
   <soapenv:Body>
      <ejb:doActionPackage>
         <arg0>
            <!-- Action -->
            <actions>
               <category>MANAGEMENT</category>
               <cmd>LOGIN</cmd>
               <parameters>
                  <name>USERNAME</name>
                  <value>iPhone</value>
    </parameters>
<parameters>
     <name>PW</name>
<value>iPhonePW</value>
</parameters>
            </actions>
            <!-- Login Parameters-->
            <parameters>
               <name>SESSION-ID</name>
               <value>iPhoneWSClient</value>
            </parameters>
         </arg0>
      </ejb:doActionPackage>
   </soapenv:Body>
</soapenv:Envelope>
```

Abbildung 4.6: Beispiel SOAP Nachricht für Login

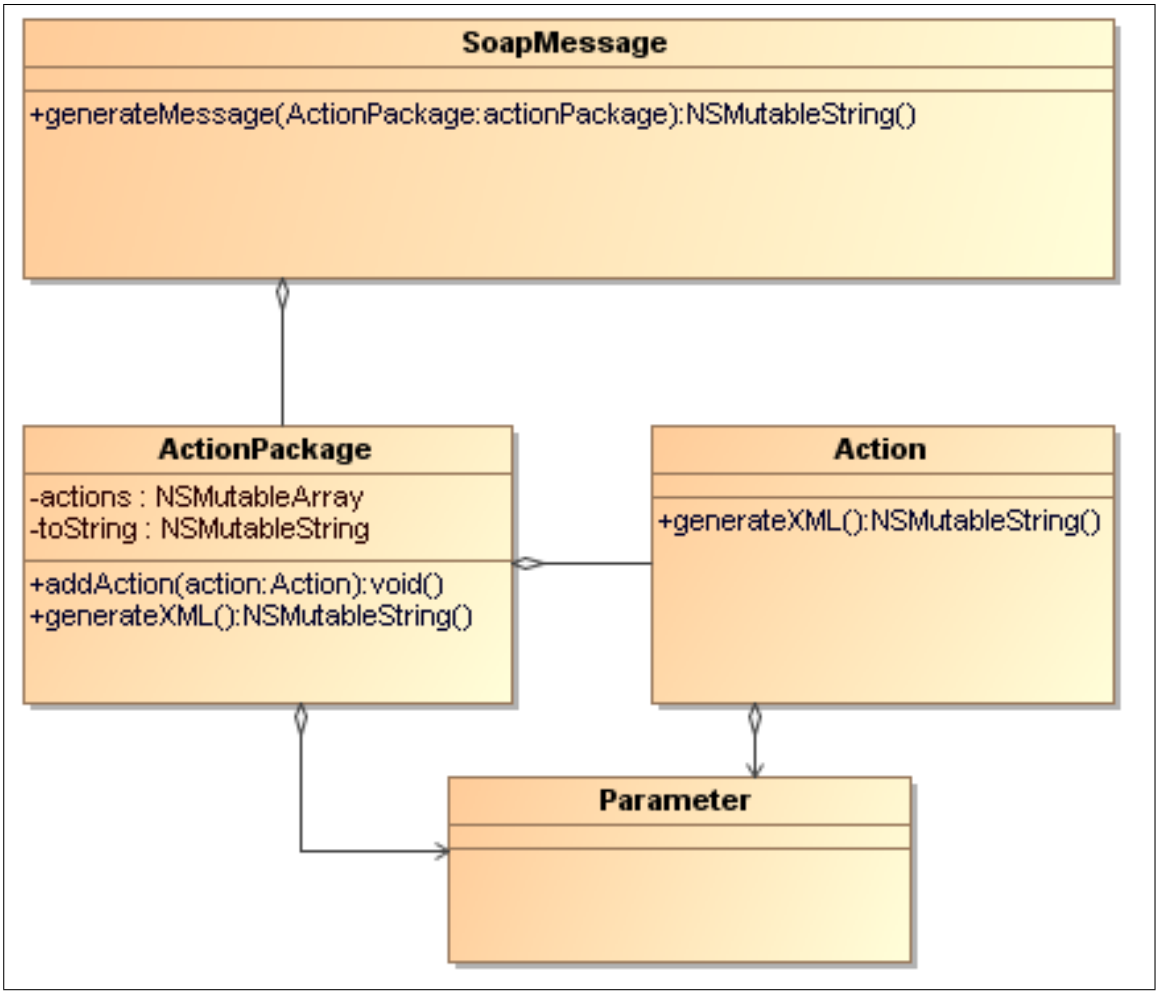

Abbildung 4.7: Datenklassen zur Nachbildung von SOAP Elementen

Hauptelement im SOAP-Body ist das ActionPackage. Ein ActionPackage beinhaltet zum einen Teilaufgaben, so genannte „Actions“, und zum anderen, als Login Parameter, die SESSION-ID des Nachrichten Senders. Es können eine oder mehrere Actions innerhalb eines ActionPackages übermittelt werden. Jede Action besteht aus den Elementen „category“, „cmd“ und einer Anzahl kommandospezifischer Parameter. Die Parameter setzten sich hierbei jeweils aus „name“ und „value“ zusammen. Abstrahiert kann man eine Action als Methodenaufruf samt Parametern auffassen. Um die SOAP Nachrichten mittels Code zu generieren, wurde die erläuterte Struktur mittels Klassen nachgebaut (siehe Abbildung 4.7). Jede der Klassen bietet die Möglichkeit, die jeweils benötigten Attribute zu speichern. Über die Methoden generateMessage() bzw. generateXML() der Klassen SoapMessage, ActionPackage und Action können rekursiv die SOAP-Nachrichten zusammengebaut werden.

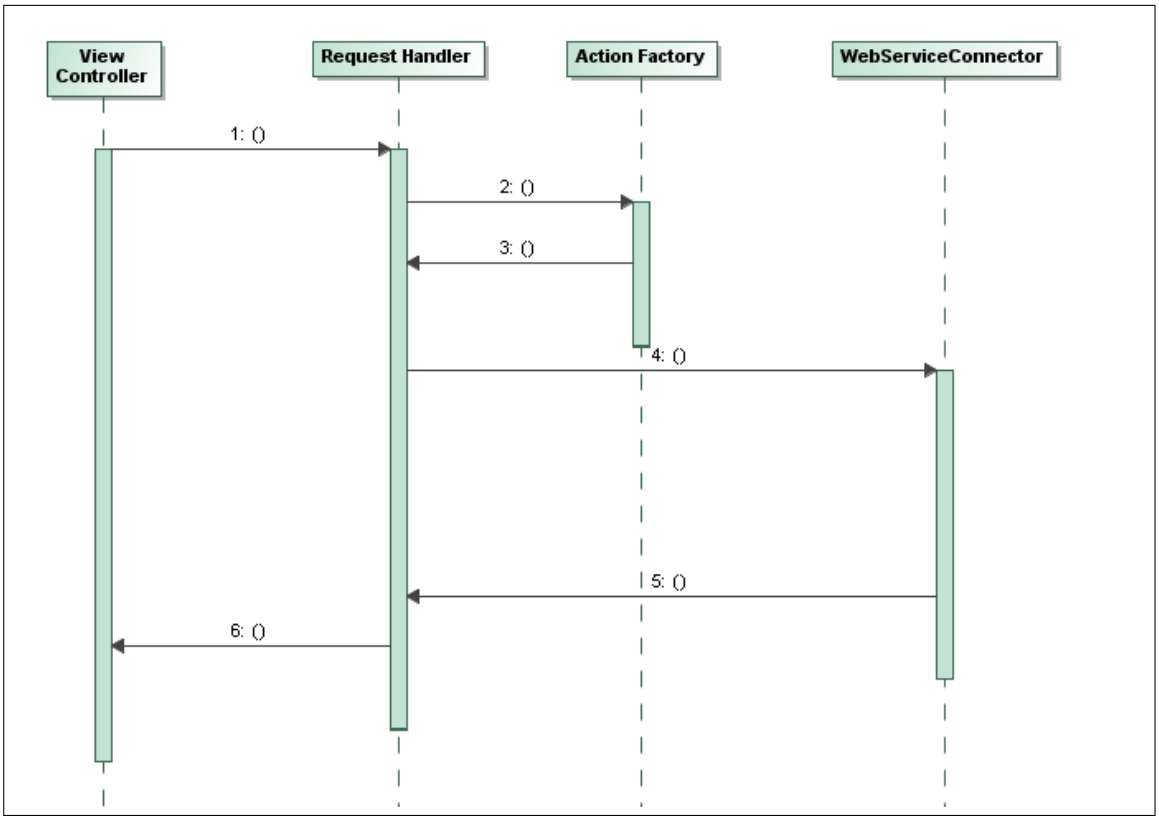

Abbildung 4.8: Clientseitiger SOAP Kommunikationsablauf

Das Senden und Empfangen von SOAP Nachrichten geschieht in mehreren Teilschritten , wie in Abbildung 4.8 dargestellt.

Kernelement der Nachrichtenerstellung ist die Klasse ActionFactory. Sie ist an das Factory Pattern angelehnt und bietet die Möglichkeit, für alle relevanten WebService-Methoden, eine entsprechende SOAP Nachricht zu generieren. Mit Hilfe der übergebenen Parameter werden die entsprechenden Elemente der SOAP Nachricht zunächst instanziiert. Anschließend wird über rekursives Aufrufen der entsprechenden Methoden (generateMessage() bzw. generateXML()), der Objekte, die XML Struktur generiert.

In der Aufrufhierachie eine Ebene höher angesiedelt, ist die Klasse RequestHandler. Der RequestHandler generiert mit Hilfe der ActionFactory eine SOAP Nachricht, welche dann mittels der Klasse WebServiceConnector an den Server gesendet wird. Zudem wird in der Klasse RequestHandler auch die Verarbeitung der Daten durchgeführt.

Die Klasse WebServiceConnector kümmert sich um die HTTP-Kommunikation. Sie sendet die generierte SOAP Nachricht an den Server und wartet auf dessen Antwort, welche dann von ihr geparst wird. Im Rahmen des Parsen werden die Elemente der Nachricht mit Hilfe der gleichen Datenklassen, wie beim Erstellen der SOAP Nachricht, in eine lokale Objektstruktur überführt. Als Rückgabe liefert Sie ein ActionPackage an den RequestHandler.

Dieser verarbeitet dann alle Actions des ActionPackages. Für jede Action wird ermittelt, um welchen „CMD“ es sich handelt und eine entsprechende Methode zur Verarbeitung der Daten aufgerufen. Im Rahmen dieser Methoden wird dann auch gegebenenfalls die lokale Datenstruktur (folgt in 4.4) an die neuen Daten angepasst.

Zusammengefasst gibt es den folgenden Kommunikationsablauf. Ausgelöst durch eine Nutzereingabe ruft ein ViewController den RequestHandler auf. Dieser generiert über die ActionFactory eine SOAP-Nachricht und versendet diese über den WebService-Connector. Der WebServiceConnector wartet auf die Antwort, parst diese und übergibt ein ActionPackage zurück an den RequestHandler. Hier werden die Daten weiterverarbeitet und gegebenenfalls die lokale Datenstruktur angepasst. Im Anschluss wir dem ViewController der Abschluss der Kommunikation mitgeteilt.

4.4 Datenhaltung

Generell gibt es bei der Applikationsentwicklung für iPhone OS verschiedenen Möglichkeiten Daten über eine Applikationslaufzeit oder auch persistent zu speichern:

- Eigene Objekt basierte Datenstruktur
- Property Lists
- SQL Lite Datenbank
- Core Data Framework

Im Falle der mobilen LASAD Anwendung, müssen die vom Server empfangenen Daten innerhalb der Application über mehrere Benutzungschritte hin verfügbar bleiben. Eine persistente lokale Speicherung der Daten über eine Ausführungssequezen hinaus ist hierbei nicht nötig. Grund hierfür ist, dass die Antwort auf die Action „JOIN MAP“ ohnehin immer den aktuellsten Datenstand der Map beinhaltet. Durch eine persistente Datenhaltung wäre also keine Minderungen des Datentransfers möglich und alle relevanten Daten werden ohnehin neu übertragen.

Ein besonders zu beachtender Aspekt bei der Gestaltungen der Datenstruktur ist durch die Verwendung von Table Views gegeben. Table Views listen üblicherweise die Daten ein oder zwei dimensionaler Arrays auf. In einem eindimensionalen Array definiert der Index die spätere Tabellenzeile. Im anderen Falle des zweidimensionalen Array bildet die erste Dimension die Tabellen Section (Gruppe) und die 2. Dimension die Tabellenzeile ab. Klickt nun der Benutzer auf eine Zeile, so erhält der Controller die Information, dass in Section X eine Zeile Nummer Y geklickt wurde. Aus diesem Grunde ist es sinnvoll,die Daten auf in einem Array oder einer anderen Datenstruktur, wo man gezielt Daten nach Section und Zeilennummer auslesen kann, bereitzustellen.

Diese, durch die Table Views geschaffene, Anforderung lässt sich mit mehreren, der zur Auswahl stehenden Technologien, realisieren. Für die Realisierung der lokalen Datenhaltung der mobilen LASAD Anwendung wurde eine eigene Objektbasierte Datenstruktur gewählt. Die Anwendungsdaten werden in mehreren Objekten gekapselt, so dass Sie nur über Accessoren aufgerufen werden können. Zunächst werden so nur begrenzte Systemressourcen verbraucht und die Benutzung von Middleware-Technologie entfällt. Sollten sich jedoch die Anforderungen ändern, ist es gegebenenfalls ohne großen Auf-

wand möglich, die eigene Objektstruktur persistent in einer SQL Lite Datenbank oder etwa eine Property List zu speichern.

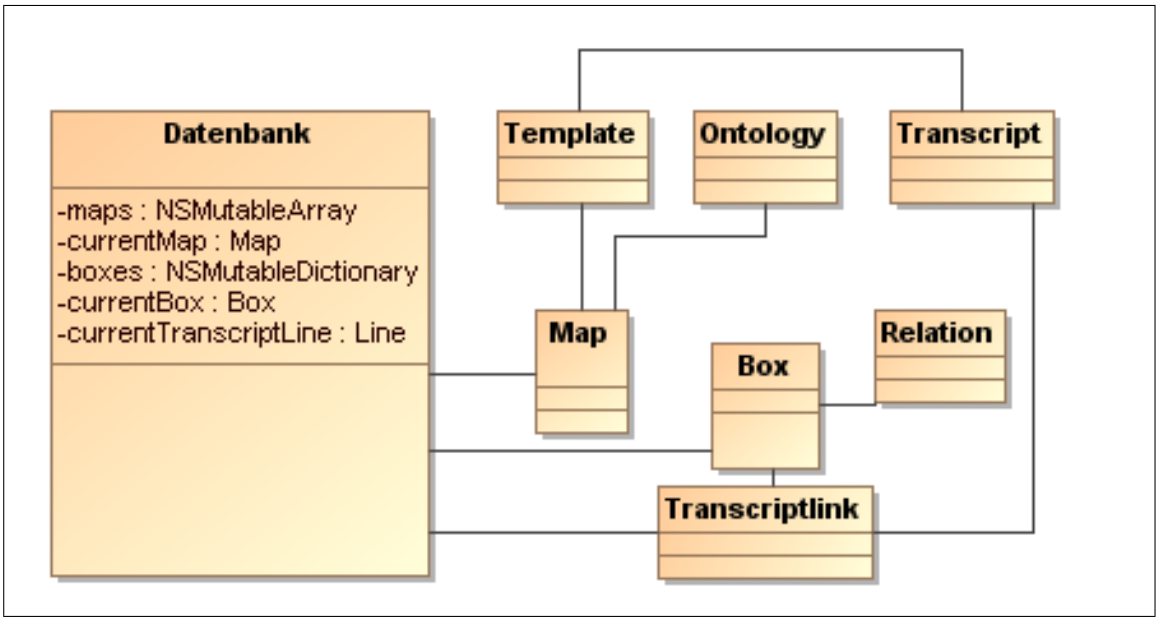

Abbildung 4.9: Datenhaltungsklassen

Die auf Objekten basierende Datenhaltung wird im Wesentlichen durch die in Abbildung 4.9 vereinfacht dargestellten Klassen realisiert. Zentrales Element ist die Klasse Datenbank. Sie implementiert das Singleton Pattern und garantiert so, dass zur Laufzeit über die Methode sharedInstance alle Aufrufe auf Daten über eine einzige Instanz laufen. Die „Datenbank“ verwaltet insbesondere eine Liste aller Maps, eine Referenz auf das Objekt der gerade ausgewählten Map, sowie die Boxen der ausgewählten Map. Diese Objekte haben entweder direkt weitere Attribute um Informationen zu speichern oder kennen untergeordnete Datenobjekte.

5 Evaluation und Fazit

5.1 Das Ergebnis

Ziel dieser Arbeit war es zu ermitteln welche Anpassungen nötig sind um argumentationsunterstüzende Systeme sinnvoll auf mobile Endgeräte zu portieren und mit Hilfe der gewonnene Erkenntnisse eine mobilen Anwendung am Beispiel von LASAD für den iPod touch zu implementieren.

Zunächst wurde ein Überblick über argumentstationsunterstüzende Systeme und mobile Endgeräte gegeben. Insbesondere wurde der Einzug der mobilen Endgeräte in das Alltagsleben und die damit verbundene steigende Bedeutung dieser Plattform herausgestellt.

Im weiteren Verlauf wurde diskutiert welche Besonderheiten bei der Entwicklung von Anwendungen für die Benutzung auf mobilen Endgeräten im Allgemeinen zu beachten sind. Darauf folgend wurde die Frage, wie sich argumentationsunterstüzende Systeme sinnvoll auf mobile Endgeräte übertragen lassen, welche insbesondere in den Abschnitten 3.2 und 3.3 erörtert wurde. Nötige Anpassungen an die Besonderheiten der mobilen Endgeräte wurden hier hinsichtlich des Funktionsumfangs und der Benutzeroberfläche ermittelt.

Nach dieser allgemeinen Betrachtung wurden die Besonderheiten der Beispielplattform iPod touch vorgestellt. Es wurde erläutert wie die Benutzeroberfläche mit Hilfe der Standartelemente von iPhone OS realisiert wurde. Des Weiteren wurde auf die Realisierung der Client-Server Kommunikation und der lokalen Datenhaltung eingegangen. Insbesondere bei der Client-Server Kommunikation war eigene Entwicklungsarbeit erforderlich, da es noch keine direkte Möglichkeit gab mittels der Standartklassen SOAP Nachrichte zu versenden.

Die im Rahmen der Arbeit entstandene Beispielanwendung für den iPod touch setzt

die in Abschnitten 3.2 und 3.3 festgehaltenen Anforderungen an Funktionsumfang und Benutzeroberfläche eine mobilen LASAD Anwendungen unter Berücksichtigung der Besonderheiten des iPod touch um.

5.2 Ansatzpunkte für weitere Schritte

Im Anschluss an diese theoretische Arbeit stellt sich die Frage, ob und wie Nutzer in der Praxis mit einer solchen mobilen Version eines argumentationsunterstüzenden Systems arbeiten. Möglicherweise lassen sich mit Probanden, die bereits die Desktop Anwendung von LASAD benutzt haben, Versuche zur Nutzung der mobilen Variante ausprobieren. In einer Studie könnte man das Nutzungsverhalten von Nutzern ohne Erfahrung im Umgang mit dem LASAD System gegenüber Nutzern mit Erfahrung vergleichen. Des Weiteren kann man die Nutzer hinsichtlich zu verbessernder oder fehlender Funktionen befragen, sowie zu Schwierigkeiten bei der Benutzung der Software.

Für die mobile LASAD Anwendung sollte die grafische Darstellung der Map weiter ausgebaut werden. Es muss unter Berücksichtigung der mobilen Plattform möglich sein über die grafische Darstellung einen schnellen Überblick über die Daten zu bekommen. Des Weiteren muss bei der Weiterentwicklung der LASAD Anwendungen überprüft werden, in wie weit Neuerungen oder Änderungen auch auf die mobile Anwendungen übertragen werden sollen oder gar müssen.

Literaturverzeichnis

1 LOLL, F. ; PINKWART, N. ; SCHEUER, O. ; MCLAREN, B. M.: Ein generisches Framework zur Erstellung von argumentationsunterstützenden Systemen. In: *Tagungsband der Multikonferenz Wirtschaftsinformatik (MKWI) 2010*, URL `http://webdoc.sub.gwdg.de/univerlag/2010/mkwi/03_anwendungen/e-_und_m-learning_wissensmanagement/02_ein_generisches_framework_zur_erstellung_von_argumentationsunterstuetzenden_systemen.pdf`. – Zugriffsdatum: 7. April 2010, 2010, S. 1427–1438

2 ZOBEL, Jörg: *Mobile Business und M–Commerce – Die Märkte der Zukunft erobern*. Hanser Fachbuch, 2001. – ISBN 3-446-21618-9

3 INC, Apple: *iPhone Application Programming Guide*. 2010-03-24. Cupertino, CA 95014: , 03 2010. – URL `http://developer.apple.com/iphone/library/documentation/Conceptual/Conceptual/MobileHIG/Introduction/Introduction.html`

4 SCHEUER, Oliver ; LOLL, Frank ; PINKWART, Niels ; MCLAREN, Bruce M.: Computer-supported argumentation: A review of the state of the art. In: *International Journal of Computer-Supported Collaborative Lerning* 5 (2010), March, Nr. 1, S. 43–102

5 DFKI, TUC: *LASAD Project Summary*. – URL `http://cscwlab.in.tu-clausthal.de/lasad/index.php?option=com_content&view=article&id=5&Itemid=5`. – Zugriffsdatum: 3. Mai 2010

6 TUROWSKI, Klaus ; POUSTTCHI, Key: *Mobile Commerce Grundlagen und Techniken*. Springer, 2004. – ISBN 3-540-00535-8

7 INC., Gartner: *Gartner Says Worldwide Mobile Phone Sales to End Users Grew 8 Per Cent in Fourth Quarter 2009; Market Remained Flat in 2009*. – URL `http://www.gartner.com/it/page.jsp?id=1306513`. – Zugriffsdatum: 25. März 2010

8 CANALYS: *Majority of smart phones now have touch screens*. – URL http://www.canalys.com/pr/2010/r2010021.html. – Zugriffsdatum: 25. März 2010

9 BUNDESVERBAND INFORMATIONSWIRTSCHAFT, Telekommunikation und neue Medien e.: *Smartphones erobern den Massenmarkt*. – URL http://www.bitkom.org/de/presse/49896_62420.aspx. – Zugriffsdatum: 25. März 2010

10 INC., Apple: *iPod Product Information*. – URL http://www.apple.com/de/ipodtouch/what-is/ipod.html. – Zugriffsdatum: 15. April 2010

11 INC., Apple: *iPod Product Information*. – URL http://www.apple.com/de/ipodtouch/specs.html. – Zugriffsdatum: 5. Mai 2010

12 APPLE INC.: *Apple iPhone OS 4 Event*. 04 2010. – URL http://events.apple.com.edgesuite.net/1004fk8d5gt/event/. – Zugriffsdatum: 14. April 2010

13 LEHNER, Franz: *Mobile und drahtlose Informationssysteme - Technologien Anwendungen Märkte*. Springer, 2003. – ISBN 3-540-43981-1

14 ETTELBRÜCK, Bernd ; HA, Sung: Mobile Marketing - Chancen und Erfolgsfaktoren des mobilen Mediums als Direktmarketing-Instrument der Zukunft. In: *E-Business, M-Business und T-Business Digitale Erlebniswelten aus Sicht von Consulting-Unternehmen*. Dr. Frank Keuper, 2003, S. 115–132. – ISBN 3-409-12026-2

15 INC, Apple: *iPhone Human Interface Guidelines*. 2010-03-24. Cupertino, CA 95014: , 03 2010. – URL http://developer.apple.com/iphone/library/documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html

16 HÖSS, O. ; SPATH, D. ; WEISBECKER, A.C. ; U., Rosenthal ; VEIT, M.: Ein Klassifikationsschema für die Architektur von mobilen Anwendungen - Erläutert an einem Praxisbeispiel zu mobilen Erfassung von Führerschein prüfungen. In: *Mobile Business – Processes, Platforms, Payments, Proceedings zur 5. Konferenz Mobile Commerce Technologien und Anwendungen (MCTA 2005) Universität Augsburg, 31.01.– 02.02.2005. GI 2005.*, Gesellschaft für Informatik, 2005, S. 131–142. – URL http://subs.emis.de/LNI/Proceedings/Proceedings59.html. – Zugriffsdatum: 24. April 2010

17 BLIEMEL, Friedhlem ; FASSCOTT, Georg: *Kundenfokus im Mobile Commerce: Anforderungen der Kunden und Anforderungen an die Kunden*. Günter Silberer and Jens Wohlfahrt and Thorsten Wilhelm, 2002. – ISBN 3-409-11905-1

18 APPLE INC.: *Apple - Apps for iPad*. 04 2010. – URL `http://www.apple.com/ipad/apps-for-ipad/`. – Zugriffsdatum: 20. April 2010

19 APPLE INC.: *iPhone Dev Center*. – URL `http://developer.apple.com/iphone/index.action`. – Zugriffsdatum: 5. Mai 2010

Anhang

Anhang 1: Ehrenwörtliche Erklärung

E R K L Ä R U N G

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäßaus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind, und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Ort, Datum Unterschrift der Kandidatin/des Kandidaten