Hochschule für Angewandte Wissenschaften Hamburg
*Hamburg University of Applied Sciences*

# Masterarbeit

Sven Vollmer

## Konzeption und Realisierung einer Usability-Komponente im Kontext von Android

*Fakultät Technik und Informatik*
*Department Informatik*

*Faculty of Engineering and Computer Science*
*Department of Computer Science*

Sven Vollmer

# Konzeption und Realisierung einer Usability-Komponente im Kontext von Android

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Olaf Zukunft
Zweitgutachter : Prof. Dr. Stefan Sarstedt

Abgegeben am 13. Juli 2010

**Sven Vollmer**

## Thema der Masterarbeit

Konzeption und Realisierung einer Usability-Komponente im Kontext von Android

## Stichworte

Android, Usability, Smartphone

## Kurzzusammenfassung

Die Masterarbeit handelt von der Konzeption und prototypischen Realisierung einer Usability-Komponente im Kontext von Android. Mit Hilfe der Usability-Komponente soll der Entwickler die Möglichkeit haben, Rückschlüsse auf die Usability seiner Anwendung zu erhalten. Anhand der Analyse wird aufgezeigt, welche Anforderungen an solch eine Usability-Komponente gestellt werden. Auf Basis der Anforderungen werden im Design ein Entwurf für eine Usability-Komponente erstellt und realisiert. Zusätzlich wurden Entwürfe für das mobile Endgerät und den Usability-Server vorgestellt, welche den Einsatz der Usability-Komponente auf beiden zeigen. Abschließend findet eine Bewertung der Arbeit mit Hilfe eines Anwendungsbeispiels statt.

**Sven Vollmer**

## Title of the Masterthesis

Design and Implementation of an Usability Component in context of Android

## Keywords

Android, Usability, Smartphone

## Abstract

This thesis deals with the design and implemantation as proof of concept of an usability component in the context of the Android mobile platform. It is intended for use as a proof-of-concept prototype. By integrating this component into their applications, developers benefit from possibilities to monitor various usability aspects. Following an analysis has been performed to identify the requirements for an usability component and what is needed to fulfill them. On the basis of these requirements of a design an implementation of the usability component has been developed. Additionally designs for the mobile device and the usability server were presented, which shows the employment of the usability component to both. At the end, an evaluation will be performed with the help of an application example.

# Inhaltsverzeichnis

# 1 Einführung

Mobile Anwendungen nehmen durch die Smartphone Betriebssysteme iPhoneOS (IPhone) und Android an Bedeutung zu. In der Studie Gartner (2010) sind die Zahlen der beiden Betriebsysteme im Vergleich 2008 zu 2009 weiter gestiegen. Außerdem zeigen die Downloadzahlen auf den verschiedenen Plattformen wie AppStore (IPhone) oder Android Market stetig weiter nach oben [IT-Times (2009), ZDNet (2009)]. In kleinen Entwicklerteams werden mobile Anwendungen für das iPhone von Apple oder Android von Google entwickelt. Später werden diese auf den Plattformen gegen eine Gebühr oder kostenlos angeboten. Die Plattformen übernehmen hierbei den Vertrieb. Die Entwickler zahlen im Gegenzug bei jedem Verkauf einen Teil als Gebühr. Bei dem Open Source Projekt Android von Google handelt es sich um ein Linux-Betriebsystem mit einer Java Virtual Machine und einem umfangreichen Framework. Mit Hilfe dieses Frameworks lassen sich vielseitige Applikationen entwickeln. Der Open Source Charakter ermöglicht eine Vielfalt von Möglichkeiten bei der Erstellung neuer Applikationen.

## 1.1 Motivation

Neuartige Bedienungskonzepte, wie z.B. das Touch-Display machen die zunehmende Rechenpower und Funktionsvielfalt der mobilen Endgeräte für den breiten Konsumermarkt attraktiver. Ein weiterer Faktor sind die schnelle Datenübertragung per UTMS/EDGE und die immer niedriger werdenen Gebühren für das mobile Surfen. Die Betriebssysteme für die mobilen Endgeräte, insbesondere Android oder IPhoneOS, bieten mit ihrem Markets/Stores die Möglichkeit, für Enwickler ihre Anwendungen dem breiten Konsumermarkt anzubieten. Vieler dieser Entwickler haben keine großen Usability Labors oder möchten einfach nicht viel Zeit in Usability-Untersuchungen investieren. Die Stores/Markets bieten zwar die Option die Applikation zu bewerten, leider reicht dieses nicht aus, um Probleme in der Bedienung oder beim Ablauf der Anwendung zu erkennen. Der Entwickler bräuchte eine Möglichkeit, in seine Anwendung eine Art von Usability-Komponente zu integrieren. Hiermit sollen Probleme in der Bedienung oder im Ablauf der Anwendung erkannt werden. Außerdem kann mit der Usability-Komponente die Applikation von vielen Benutzern getestet werden. Hiermit erhält der Entwickler genügend viele Daten, um Probleme in seiner Applikation zu erkennen. Diese Daten müssten dann aber für den Enwickler sinnvoll aufbereitet werden.

## 1.2 Zielsetzung

Aus der Motivation heraus soll im Rahmen dieser Arbeit eine Usability-Komponente für Android-Applikationen entwickelt werden. Die Usability-Komponente soll leicht in eine bestehende oder noch zu entwicklende Applikation eingebunden werden. Im theoretischen Teil wird analysiert, welche Anforderungen es an Usabilty im mobilen Umfeld gibt. Ein wichtiges Ziel ist hierbei die Bestimmung der messbaren Größen. In der Analyse werden die wichtigen funktionalen und nicht-funktionalen Anforderungen an den Entwurf aufgestellt. Auf dessen Grundlage wird ein Prototyp mit Hilfe einer komponentenbasierten Architektur realisiert. Dieser Prototyp wird in ein Anwendungsbeispiel integriert und deren Tragfähigkeit dadurch getestet.

## 1.3 Aufbau der Arbeit

Die Arbeit ist in sieben Kapitel unterteilt. In Kapitel 2 werden alle Gundlagen beschrieben und definiert, die für diese Arbeit wichtig sind. Hierzu werden in Kapitel 2 die Themengebiete Usability , *Methoden der Usability*, *Android* und *Usability und Android* beleuchtet. Im Kapitel 3 folgt die Analyse. In der Analyse wird mit Hilfe eines Anwendungsbeispiels die funktionalen und nicht-funktionalen Anforderungen herausgearbeitet. Anschließend wird in Kapitel 4 auf Grundlage der Analyse das Design vorgestellt. Im Design wird die Systemarchitektur 4.1, Anwendungsarchitektur 4.2 und der Feinentwurf 4.3 beschrieben. Die Realisierung wird in Kapitel 5 erläutert. Dabei werden die Details des Prototyps angesprochen. Als Nächstes wird das Anwendungsbeispiel mit zwei Szenarien beschrieben. Anhand jener wird die Integration des Prototyps in das Anwendungsbeispiel beschrieben. In Kapitel 6 wird eine Bewertung der Arbeit vorgenommen. Hierbei werden die Anforderungen mit dem Design und der Realisierung abgeglichen. Die Ergebnisse werden dabei kritisch betrachtet. Zu guter Letzt wird im letzten Kapitel 7 die Arbeit zusammengefasst und ein Ausblick gegeben.

# 2 Grundlagen

Dieses Kapitel beinhaltet die notwendigen Grundlagen für diese Arbeit. Der Abschnitt 2.1 behandelt das Themengebiet Usability. Hiernach werden die unterschiedlichen Methoden der Usability (s. Abschnitt 2.2) beschrieben und die Unterschiede aufgezeigt. Die Umsetzung erfolgt mit der Android Plattform (s. Abschnitt 2.3). Im Abschnitt 2.4 werden die genutzen Programmierwerkzeuge vorgestellt. Zum Schluß werden existierende Arbeiten in den Themengebieten Android und Usability aufgezeigt.

## 2.1 Usability

Im folgenden Abschnitt 2.1 wird der Begriff Usability genauer definiert und deren vielfältige und unterschiedliche Bedeutung gezeigt. Insbesondere werden hier die wichtigen Elemente und Schlussfolgerungen von Usability herausgefiltert.

Der Begriff Usability kommt aus dem englischen Sprachraum und bedeutet übersetzt Brauchbarkeit, Verwendbarkeit, Benutzerfreundlichkeit, Gebrauchstauglichkeit, Nutzbarkeit oder Benutzbarkeit [dict.cc]. Usability ist aber als solcher Begriff im deutschen Sprachraum im Gebrauch, da die oben genannten Begriffe in ihrer einzelnen Bedeutung nicht den Kern von „Usability“ treffen. Vielmehr ist es die Menge aller Begriffe, die Usability beschreiben. [Schuhmacher]

### 2.1.1 Definition

Die Internationale Organisation für Standardisierung legte in der ISO 9241 folgendes fest:
*“Usability eines Produktes ist das Ausmaß, in dem es von einem bestimmten Benutzer verwendet werden kann, um bestimmte Ziele in einem bestimmten Kontext effektiv, effizient und zufriedenstellend zu erreichen.“* [ISO 9241, zit. nach Eichinger]

Nielsen sagt folgendes über Usability:
*“Usability is the measure of the quality of the user experience when interacting with something – whether a Web site, a traditional software application, or any other device the user can operate in one way or another.“* [Jakob Nielsen (1998), zit. nach Eichinger]

Nielsen beschreibt hier die Interaktion des Benutzers mit irgendetwas, welches in der Regel computerbezogene Themen wie Webseiten, Software-Applikationen auf Computern, Handys, Videorekordern und anderen elektronischen Geräten bezieht; in diesem Zusammenhang ist es genau das Maß an Qualität dieser Interaktion.

*"Usability really just means making sure that something works well: that a person of average ability and experience can use the thing for its intended purpose without getting hopelessly frustrated."* [Krug (2000)]

Die verschiedenen Definitionen zeigen, wie komplex der Begriff Usability ist. Es geht nicht nur um eine vereinfachte Interaktion für den Benutzer, sondern vielmehr darum, den User zu verstehen und ihn entsprechend in seinen Aufgaben und Zielen zu unterstützen. Wie bereits erwähnt, legte Eichinger die Elemente wie Effizienz, Effektivität und Zufriedenheit fest [Eichinger]. Hinzukommen kommen aber noch Fehlertoleranz und Lernbarkeit. Diese Elemente nennt man auch Usability-Attribute, welche im Abschnitt 2.1.2 genauer beschrieben werden.

### 2.1.2 Usability-Attribute

In den folgenden Abschnitten werden die Usability-Attribute (angelehnt an [Erdinger]) beschrieben. Sie haben je nach Software-Applikation immer eine unterschiedliche Gewichtung. Außerdem hängt es stark von der Art der Software ab, ob beispielsweise wichtige Transaktionen damit ausgeführt werden können oder sie mehr einen spielerischen Charakter besitzt. Jeder Entwickler muss vorher festlegen, mit welcher Gewichtung die Attribute belegt werden bzw. welche es in Zukunft zu verbessern gilt.

#### Effektivität

Effektivität bezeichnet die Genauigkeit und Vollständigkeit, mit der ein Ziel erreicht wurde. Der Benutzer soll vor allem bei der Erfüllung des Ziels unterstützt werden. Gemessen werden kann, wie gezielt und vollständig der Benutzer die Aufgabe lösen wird. Hier gibt es die Möglichkeit, den Pfad der Navigation zu verfolgen, um die Komplexität oder Probleme in der Navigation zu erfassen. Außerdem können vom Benutzer gewählte Abkürzungen in späteren Entwicklungen auch für die anderen Benutzer nutzbar gemacht werden.

#### Effizienz

Der Benutzer sollte die Aufgabe möglichst in kurzer Zeit erledigen können. In Software-Applikationen kann man hierzu die Dauer erfassen, welche benötigt wird, die Aufgabe zu

erfüllen. Auch die Anzahl von Mausklicks können eine Aussage über die Effizienz machen. Hier geht es um das schnelle und klare Navigieren des Benutzers. Beachten sollte man auch die Erfahrungen des Benutzers. Erfahrenen Benutzern sollten Abkürzungen zum Beschleunigen der Aufgabenerfüllung zur Verfügung stehen. Des Weiteren sollten vorhandene oder zukünftige Abkürzungen den Anfänger nicht irritieren bzw. überfordern.

#### Zufriedenheit

Die Zufriedenheit des Benutzers bei der Erledigung seiner Aufgabe ist von besonderer Bedeutung. Das Design und die Struktur sollten für den Benutzer während der Navigation durch die Software-Applikation als angenehm empfunden werden. Hierzu zählen einfache und klare Navigation, Textlänge/-aufbereitung, Farben, sinnvolle Platzierung von Grafiken bzw. Bildern. Zufriedenheit ist ein sehr subjektives Empfinden. Die Entwickler sollten sich an die entsprechende Guidelines halten. Messbar ist die Zufriedenheit nur durch Gespräche mit dem Benutzer oder eine vom Benutzer abgegebene Bewertung.

#### Fehlertoleranz

Im Idealfall sollten keine schwerwiegenden Fehler in der Software-Applikation auftreten. Diese lassen sich heutzutage nicht gänzlich ausschließen. Beim Auftreten von Fehlern oder gewollte Ausnahmen (Exceptions) sollte der Benutzer bei der Bewältigung unterstützt werden. Fehler sollten daher eine klare und verständliche Beschreibung enthalten und dem Benutzer eine Hilfe geben, diesen zu beheben. Die Häufigkeit von Fehlern und besonders das Verhalten der User hierbei können gemessen werden.

#### Lernbarkeit

Die Software-Applikation sollte für den Anfänger leicht zu erlernen sein. Leichte und verständliche Navigation müsste das Ziel sein. Zum Lösen einer Aufgabe kann die Komplexität der Navigation mit Hilfe eines Graphs, der beim Durchlaufen der Software-Applikation entsteht, dargestellt werden. Diese kann dann durch entsprechende Algorithmen gemessen werden.

### 2.1.3 Usability-Engineering

Usability nimmt heutzutage eine immer größere Bedeutung ein. Hieraus entwickelte sich in der Wissenschaft das Usability Engineering. Eichinger beschreibt Usability-Engineering als

einen fortlaufenden Prozess, mit dessen Hilfe bei der Entwicklung eines Produktes Usability definiert, gemessen und verbessert werden kann [Eichinger].

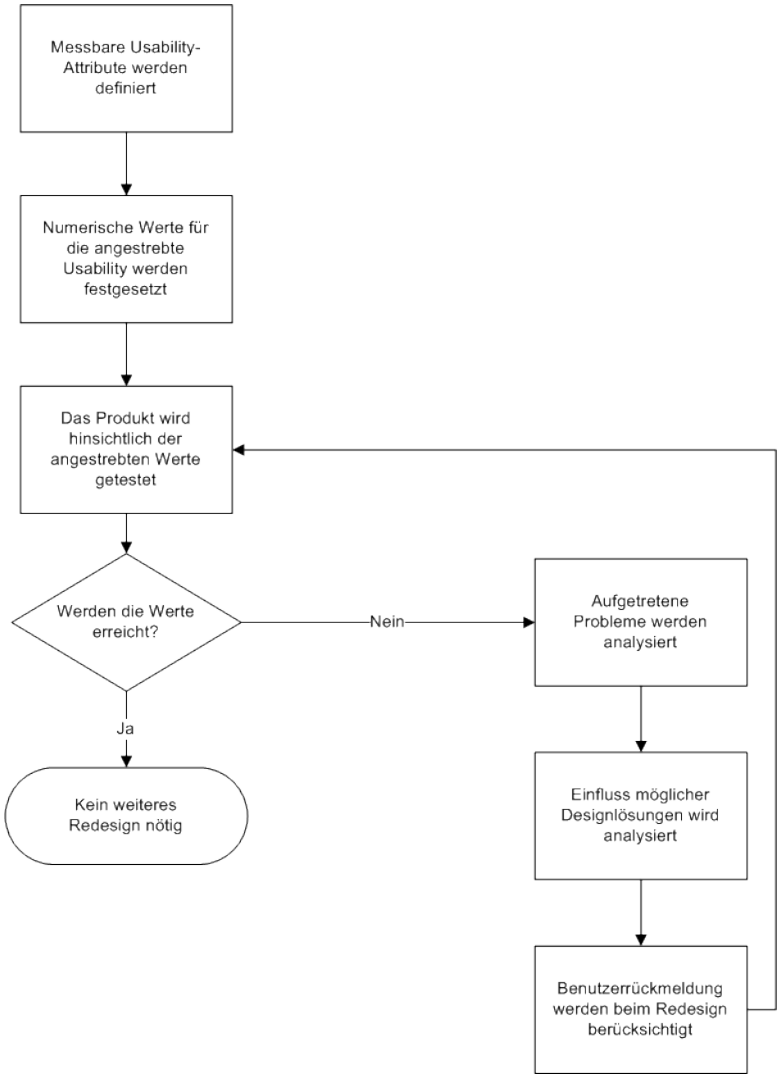



Abbildung 2.1: Usability-Engineering Prozess [Eichinger]

### Usability-Ziele

1. Benutzerbeschreibung:
   Die Benutzerbeschreibung soll die verschiedenen Benutzergruppen widerspiegeln. Die Unterteilung erfolgt durch Eigenschaften wie Alter, Computererfahrungen, Erfahrungen in der Domäne [1] der Software. Außerdem wäre eine Unterscheidung in Kategorien wie Anfänger und Experte sinnvoll, da diese auch Einfluss auf die Usability-Tests haben.

   Es gibt Produkte, die von vornherein auf Gelegenheitsnutzer zugeschnitten sind, wie z.B. Informationssysteme an Bahnhöfen. Die Kategorie Gelegenheitsnutzer beschreibt User, die keine Erfahrungen mit solchen Systemen besitzen, und ihnen zum anderen die Zeit fehlt, die Bedienung des Systems ausführlich zu erlernen. Es ist also von großer Bedeutung, bei den Zielen die relevanten Eigenschaften der Benutzergruppen festzulegen und diese während des Prozesses zu berücksichtigen.

2. Aufgabenanalyse:
   Das Ziel der Aufgabenanalyse ist es, die Hauptaufgaben des Benutzers zu erfassen und deren Häufigkeit zu bestimmen. Hierbei werden verschiedene Techniken eingesetzt wie z.B. Beobachten und Befragungen des Benutzers. Das frühe und exakte Erfassen der Aufgaben hilft bei der Verbesserung der Usability.

3. Usability-Attribute:
   Die Usability-Attribute sind in Kapitel 2.1.2 schon genauer beschrieben. Diese müssen weiter in Unterkategorien gegliedert werden. Sie sind notwendig, um sie in messbare Größen zu verwandeln. Hierzu gehören zum Beispiel Fehlerraten, Flexibilität oder Erlernbarkeit. In den Zielen werden zu den jeweiligen Messgrößen kritische Werte festgelegt, die bei der Entwicklung als auch beim fertigen Produkt nicht überschritten werden sollten.

4. Messinstrumente:
   Ein Messinstrument dient zur Erfassung der Werte für die jeweiligen Usability-Attribute bzw. der dafür notwendigen Messgrößen. Die können sowohl subjektive (z.B. Zufriedenheit) als auch objektiver (z.B. Fehlerraten, Zeitaufwand) Art sein. Typische Messinstrumente sind Software zur Protokollierung, Videos und Fragebögen.

5. Messgrößen:
   Hier werden die relevanten Messgrößen aufgeführt und deren spezifische Gewichtung.

[1] Erfahrungen in der Domäne bedeutet in dem Zusammenhang wie viel Erfahrungen die Benutzer mit ähnlicher Software besitzen

6. Kritisches Level:
   Mit dem kritischen Level werden Grenzen für die quantitativen Ausprägungen der Attribute bzw. Messgrößen festgelegt. Sie können dabei absolute Werte annehmen wie z.B. „Ein Benutzer soll zum Lösen einer Aufgabe nur maximal zwei Fehler machen“ oder relative Werte wie z.B. „Der Benutzer soll weniger Fehler beim Lösen der Aufgabe bei Produkt A machen als bei Produkt B“. Die kritischen Werte kommen aus vorhergehenden Usability-Tests, von Experten, Feldstudien oder Marktforschungsdaten.

7. Usability-Probleme:
   Mit diesem Ziel werden Probleme aufgezeichnet, die beim Beobachten des Benutzers entstehen. Hat der User Probleme mit der Interaktion? Läuft die Bedienung und die Navigation reibungslos bzw. wo treten Probleme auf? Diese werden als qualitative Daten bezeichnet. Die Erhebung unterliegt subjektiven Einflüssen und bedarf einer Interpretation dieser Daten.

(angelehnt an [Eichinger][Nielsen (1994)])

### Usability-Tests

Es folgt eine kurze Beschreibung wie ein Usability-Test abläuft. Auf eine detaillierte Beschreibung wird verzichtet, da in dieser Arbeit nicht die Durchführung von Usability-Tests, sondern die technologische Unterstützung der Durchführung behandelt wird. Ein Usability-Test besteht aus sechs Phasen:

1. Entwicklung des Testplans:
   Der Testplan legt u.a. die Benutzergruppen fest, die zu lösenden Aufgaben, die Art der Testumgebung und den Umfang der Datensammlung.

2. Auswahl und Anwerbung der Benutzer:
   Aus festgelegten Benutzergruppen im Testplan werden die Benutzer angeworben, die auf das Profil passen.

3. Vorbereitung der Testmaterialien:
   Die Testmaterialien müssen vorbereitet werden. Hier werden z.B. die Aufgaben und Testprotokolle beschrieben, welche Fragen dem Benutzer vor und nach dem Test gestellt werden oder auch die Checkliste für den Ablauf.

4. Pilottest:
   Im Pilottest wird der Test komplett intern durchgeführt, um Probleme zu erkennen und

entsprechend zu korrigieren. Hierzu gehören zum Beispiel unklare Instruktionen, unrealistische Zeiteinschätzung, irreführende Fragen oder andere Probleme beim Lösen der Aufgabe.

5. Realtest:
   Dieser beinhaltet den realen Test mit den Benutzern.

6. Analyse und Auswertung:
   Abschließend werden die Ergebnisse analysiert und für eine Auswertung entsprechend aufgearbeitet. Die berechneten Messgrößen Median, Mittelwert und Standardabweichung werden mit den kritischen Werte verglichen. Es kann auch die Prozentzahl der Benutzer aufgeführt werden, die den Test bestanden haben. Außerdem müssen die Interviews entsprechend ausgewertet werden. Ziel sollte es sein, die Bereiche herauszufiltern, in denen es zu Problemen gekommen ist, und jene für die Nachbesserung nach Art und Wichtigkeit zu kategorisieren.

(siehe [Keith (2009)])

### 2.1.4 Usability bei mobilen Endgeräten

Der Umgang mit mobilen Endgeräten stellt besondere Herausforderungen an die Durchführung von Usability-Tests. Aus den Studien und Guidelines in Kapitel 2.5 wurden folgende Usability-Probleme bei mobilen Endgeräten identifiziert:

- Auftreten von Fehlern
  Auftretende Fehler sollen in einem klaren verständlichen Text ausgedrückt und dem Benutzer hierfür Lösungsmöglichkeiten aufgezeigt werden. Häufige Fehler können auf Verständnisprobleme in der Anwendung hinweisen.

- Probleme beim Lösen von Aufgaben
  Mit einer mobilen Anwendung kann der Benutzer verschiedene Aufgaben erfüllen. Vollendet ein Benutzer die Aufgabe des öfteren nicht, könnte das auf Bedienungsprobleme hinweisen. Dies kann ein Indiz für Schwierigkeiten mit der Aufgabe oder einer zu komplexen Navigation sein

- Bedienungsfehler
  Das häufige Betätigen des Touch-Displays oder Tasten des Android-Handys können in einer Anwendung auch auf Bedienungsprobleme hindeuten.

Die Erfassung ist nicht vollständig. Weitere Untersuchungen könnten weitere Usability-Probleme zu Tage fördern.

## 2.2 Methoden der Usability

Für die Durchführung eines Usability-Tests gibt es verschiedene Methoden: Im folgenden Abschnitt werden zwei vorgestellt, zum einen die Laborstudien in Abschnitt 2.2.1 und zum anderen die Feldstudien in Abschnitt 2.2.2. Diese Methodiken haben jeweils ihre Stärken und Schwächen, wobei es hier stark auf die Art der Anwendung ankommt, insbesondere im Zusammenspiel mit dem Kontext des Benutzers. Abschließend werden beide in in Kapitel 2.2.3 miteinander verglichen und die Bedeutung des Kontextes erklärt.

### 2.2.1 Laborstudien

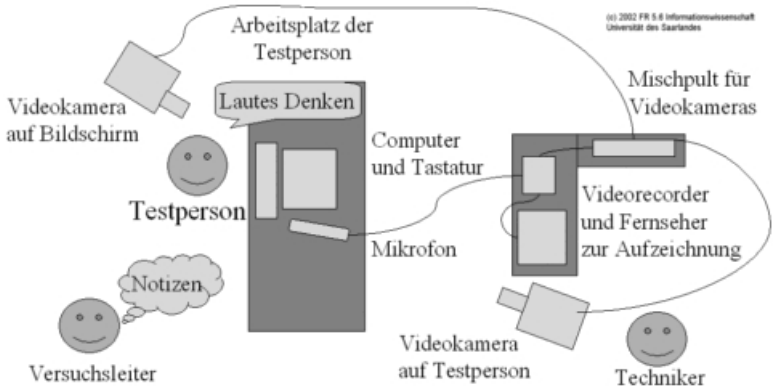



Abbildung 2.2: Aufbau eines Usability-Test im Labor [Saarland]

In Abb. 2.2 wird der klassische Aufbau eines Usability-Tests am PC im Labor abgebildet. Der Benutzer sitzt an einem künstlich geschaffenen Arbeitsplatz und versucht, die ihm gestellten Aufgaben zu erfüllen. Das Laut-Denken wird während des Tests mit Hilfe der Mikrofone aufgenommen. In dem so genannten Regieraum befinden sich Kamera, Mikrofone und Lautsprecher, um den Benutzer zu überwachen. In der Regel sitzt der Versuchsleiter während der Durchführung im Regieraum, er kann sich aber auch in dem Versuchslaborraum aufhalten. Im Regieraum wird der gesamte Usability-Test mit Bild und Ton aufgezeichnet, um später anhand dessen Rückschlüsse ziehen zu können. Außerdem können weitere Metriken wie Tastatureingaben, Mausklicks oder zurückgelegte Mausstrecke für eine spätere Auswertung aufgezeichnet werden. Bei solch einem Usability-Test befindet sich der Benutzer in der künstlich geschaffenen Umgebung, wo er kaum Störungen und Ablenkungen unterworfen ist. Diese könnten künstlich erschaffen werden, aber würden nur annähernd dem natürlichen Kontext des Benutzers entsprechen. Der größte Nachteil einer Laborstudie, gerade im Zusammenhang bei mobilen Endgeräten, ist der fehlende natürliche Kontext. Hier sollte eine Feldstudie vorgezogen werden, falls diese nicht zu gefährlich oder unpraktisch ist. Zu diesen Ergebnissen kam Priebs [Priebs (2006)] in seiner Ausarbeitung.

### 2.2.2 Feldstudien

In einer Feldstudie befindet sich der Benutzer in seiner natürlichen Umgebung. In Abschnitt 2.2.1 wurde erwähnt, dass Feldstudien gerade bei mobilen Endgeräten vorzuziehen sind, da dort der entsprechende Kontext gegeben ist. Mobile Anwendungen können überall gestartet werden und vor allem kann es hier auch zu realistischen Störungen oder anderen Problemen kommen. Der jeweilige Kontext der Feldstudie entscheidet, ob sich Überwachungsmöglichkeiten (wie in Abschnitt 2.2.1 erwähnt) anbieten. In einem Krankenhaus würden sich hierfür zum Beispiel Möglichkeiten bieten. Bei einer mobilen Anwendung, wie beispielsweise einer Fahrplanauskunft, die praktisch überall benutzt werden kann, können jedoch keine Überwachungstechniken wie Mikrofon oder Kamera eingesetzt werden. Es wäre hier allerdings denkbar, das Mikrofon oder die Kamera des mobilen Endgerätes zu nutzen. Die Kameras bei den mobilen Endgeräte sind nicht immer zum Benutzer gerichtet und dessen Sichtfenster ist teilweise auch eingeschränkt. Der jeweilige Kontext der mobilen Anwendung entscheidet, welche Überwachungsmöglichkeiten im Usability-Test machbar sind.

### 2.2.3 Vergleich zwischen Labor- und Feldstudie

Duh u. a. stellten bei ihrer Studie fest, dass der Kontext gerade bei mobilen Endgeräten in Feldstudien andere Anforderungen und Probleme für den Benutzer entstehen läßt als im Vergleich zu einer Laborstudie: Die Feldstudien bilden zwar den natürlichen Kontext ab, jedoch sind die Überwachungsmöglichkeiten hier wiederum eingeschränkt. Hier wäre es zum Beispiel denkbar, die Hardware-Ressourcen des mobilen Endgerätes zu nutzen, um dem Benutzer beispielsweise mit dem Mikrofon die Möglichkeiten zu geben, aktuelle Kommentare zu Problemen aufzuzeichnen. Auch die Kamera könnte genutzt werden, um die Reaktion des Benutzers auf die jeweilige Situation anhand der Gesichtsmimik abzulesen. Laut der Studie von Irion sollte man die Beeinflussung durch Hemmungen vor dem Aufzeichnungsgerät möglichst gering halten. Die Benutzer sollten über den Zweck der Aufzeichnung informiert werden und durch eine entsprechende Vorlaufphase an die Situation gewöhnt werden.

In Abschnitt 2.2.1 wurde die Bedeutung des Kontextes bei mobilen Endgeräten geschildert. Man könnte annehmen, der Einsatz von Feldstudien sei denen von Laborstudien stets vorzuziehen. Es herrscht jedoch Uneinigkeit, welche Methode bevorzugt wird. Eine Untersuchung aus dem Jahre 2003 von Kjeldskov und Graham zeigte, dass 71% im Labor und 19% als Feldstudien durchgeführt werden. Als häufiger Grund wird größerer Zeitaufwand und die damit verbundenen höheren Kosten genannt. Ebenfalls Uneinigkeit herrscht über den tatsächlichen Nutzen. Anhand zweier Beispiele aus Studien wird dies verdeutlicht: In der Studie von Kjeldskov u. a. über Nutzen von Feldevaluationen wurde ein elektronisches Patienten-Aufzeichnungssystem (EPR) sowohl als Feldstudie als auch im Labor evaluiert. Das System

unterstützt Krankenschwestern bei den täglichen Aufgaben und Kontrollgängen. Für das Labor wurde ein Raum exakt nachgebaut wie er im Krankenhaus existiert. Und im Krankenhaus wurden zusätzlich entsprechende Kameras und Mikrofone installiert, um Benutzerinteraktion aufzuzeichnen und gleichzeitig Bewegungsfreiheit für den Propanden zu gewährleisten. Das Ergebnis war überraschend: Es wurden bei der Laborevaluation mehr Usability-Probleme gefunden als bei der Feldstudie.

In der Studie Duh u. a. (2006) sollten die Probanden mit ihrem Mobiltelefon verschiedenste Aufgaben erledigen. Die eine Gruppe der Probanden löste die Aufgaben im Labor und die andere den Feldtest unterwegs in der MRT Singapur (das dortige Metro-Netz). Die Probanden im Feldtest benötigten mehr Zeit zum Ausführen der Aufgaben und waren mehr negativen Eindrücken ausgesetzt, wie beispielsweise dem Geräuschpegel in der MRT oder zu wenig Privatsphäre. Durch den vermehrten Stress kam es zu einem Mehraufwand beim Lösen von Aufgaben. Die unterschiedlichen Ergebnisse zeigen, dass beim Usability-Test verschiedenste Faktoren Einfluss auf die Ergebnisse haben. Beim ersten Test könnte man die Vermutung anstellen, dass die Krankenschwestern viel entspannter in ihrem gewohnten Arbeitsumfeld waren als im Labortest. In der zweiten Studie waren die Probanden im Feldtest in einer nicht gewohnten Umgebung und die äußeren Einflüsse waren hier auch wesentlich höher. Diese Vermutungen können nicht belegt werden, da die Studien darüber keine Auskunft geben. Es bleibt aber festzuhalten, dass der jeweilige Kontext einen gewissen Einfluss auf das Ergebnis hat. Wenn es im finanziellen Rahmen liegt, sollte Usability sowohl als Laborstudie wie auch als Feldstudie durchgeführt werden. Alternativ kann im Vorfeld abgewägt werden, von welchem Test die besseren Ergebnisse erwartet werden.

## 2.3 Android

Android ist Plattform und Betriebssystem für mobile Endgeräte (Smartphones, Mobiltelefone, Netbooks). Es wurde von der Open Handset Alliance [Alliance (b)] entwickelt und wurde als Open Source Projekt unter die Apache v2 Lizenz gestellt. Es läuft bereits auf einigen Geräten von verschiedenen Herstellen, wie zum Beispiel HTC, T-Mobile (G1) oder auch Samsung. Die Anzahl der Geräte wird in Zukunft weiter zunehmen, weil die Hersteller fortwährend neue Technik auf dem Markt bringen.

### 2.3.1 Systemarchitektur

In Abb. 2.3 wird der Aufbau der Android Systemarchitektur anhand von Schichten gezeigt. Für eine einheitliche Hardwareabstraktion befindet sich auf der untersten Schicht ein für mobile Geräte optimierter Linux Kernel. Hier müssen die entsprechenden Treiber an die

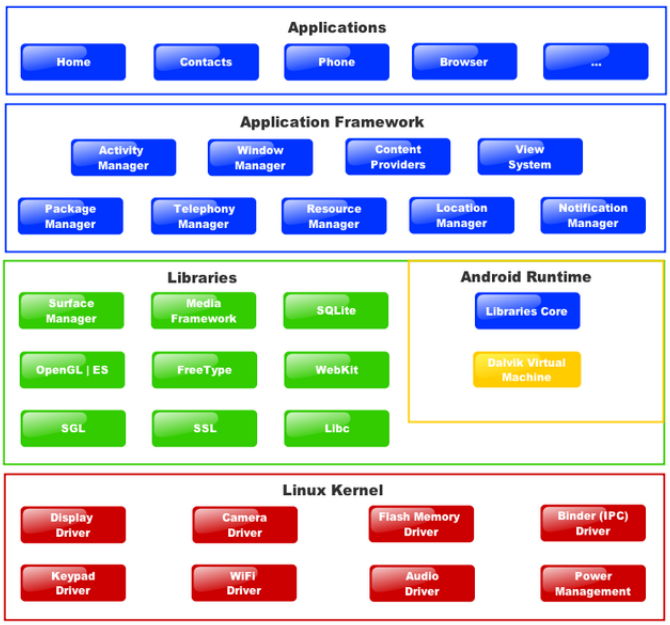



Abbildung 2.3: Android Architektur [Wikipedia (2010)]

jeweilige Technik der Hersteller angepasst werden, um Android darauf nutzen zu können. Darüber befindet sich in C geschriebene Libraries für eine performante Ausführung. Dazu gehören zum Beispiel eine HTML-Rendering-Engine oder auch die SQLite-Datenbank. Die ersten beiden Bereiche beinhalten in Java geschriebene Komponenten. Im Framework (s. Abschnitt 2.3.1) befinden sich Hilfsklassen zur Unterstützung für die Programmierung. Im Bereich *Android-Runtime* befindet sich die Dalvik VM, eine für mobile Geräte entwickelte Java Virtual Machine (JVM). Die Programme können dabei in Java geschrieben werden; es fehlen jedoch aus der Java SE Pakete, wie beispielsweise AWT oder Swing. (angelehnt an [Wixler und Schwertel (2007)][Voigt (2009)])

### Framework

In dem folgenden Abschnitt werden einige wichtige Bausteine des Frameworks für die Programmierung von Anwendungen erläutert. Dazu gehören:

- Activity:
  Activites bilden die jeweiligen, vom Benutzer bedienbarer, grafischen Oberflächen ab.
- Intents:
  Mit Hilfe der Intents können aus Anwendungen heraus Aktionen gestartet werden wie

beispielsweise der Wechsel von einer Activity zu einer anderen. Dabei müssen zwei Informationen angegeben werden und zwar die Aktion und die URI.

- Services:
  Ein Service ist ein Dienst, der im Hintergrund abläuft und der nicht direkt mit dem Benutzer interagiert.
- Content Providers:
  Mit den Content Providers wird eine Schnittstelle für Daten auf Basis von URIs bereitgestellt ebenso wie für Daten anderer Applikationen.
- Broadcast Receivers:
  Mit den Broadcast Receivern können durch externe Events wie z.B. SMS, Klingeln des Telefons oder zeitgesteuerte Prozesse aufgeweckt werden.
- Notifications und Toasts:
  Benachrichtigung des Benutzers mit Bild, Ton oder Vibrationsalarm

(angelehnt an [Wixler und Schwertel (2007)][Voigt (2009)])

### 2.3.2 Lebenszyklus einer Android-Applikation

Eine Activity kann grundsätzlich drei Zustände annehmen:

- activ oder running:
  Activity befindet sich im Vordergrund und hat den Fokus.
- paused:
  Activity ist noch sichtbar, aber wird teilweise verdeckt und hat keinen Fokus.
- stopped:
  Activity ist nicht mehr sichtbar

Die Zustandsänderungen in einer Activity werden mit den Methoden *onCreate()*, *onRestart()*, *onStart()*, *onResume()*, *onPause()*, *onStop()* und *onDestroy()* vorgenommen, die bei Bedarf überschrieben werden können.

Der Lebenszyklus (s. Abb 2.4) der Applikation startet bei *onCreate()* und endet bei *onDestroy()*. Beim Start der Activity läuft diese entweder aktiv im Vorder- oder im Hintergrund und durchläuft dabei die Methoden *onStart()* und *onStop()*. Im Lebenszyklus wird die Methode *onResume()* aufgerufen, wenn die Activity im Vordergrund aktiv ist, kann sie mit dem Benutzer interagieren. Verliert die Activity den Fokus oder wird durch andere Activities teilweise verdeckt, wird die Methode *onPause()* durchlaufen. In dem Fall einer kompletten Verdeckung durch eine andere Activity wechselt sie im Lebenszyklus in den Zustand *onStop()*.

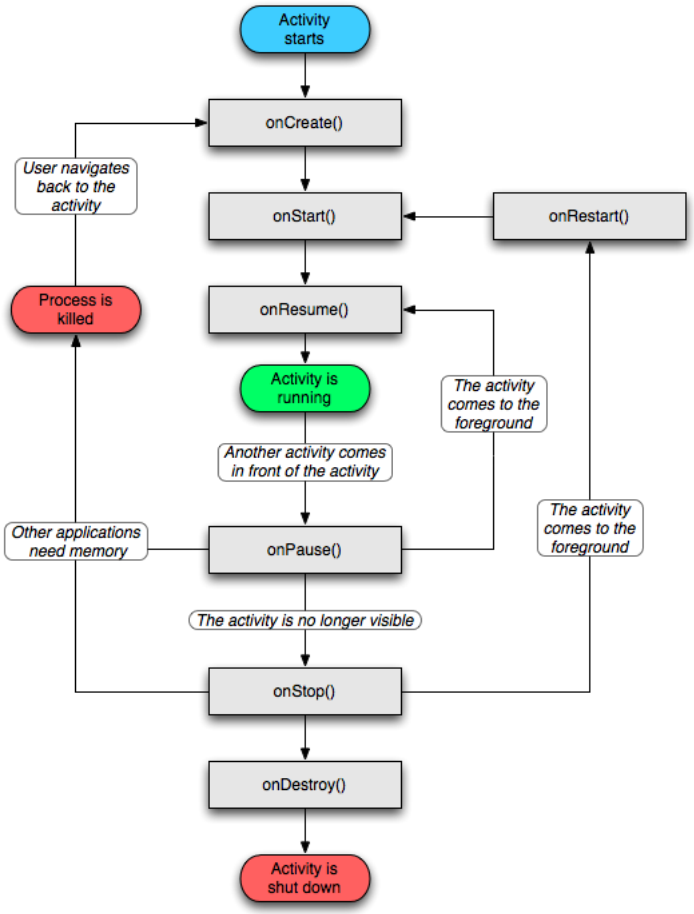



Abbildung 2.4: Lebenszyklus einer Android Applikation [Alliance (a)]

## 2.4 Programmierwerkzeuge

In diesem Abschnitt werden die Programmierwerkzeuge, die in der Arbeit zum Einsatz kommen, kurz vorgestellt.

- Eclipse und Java:
  Als Entwicklungsumgebung für den Prototyp und dem Anwendungsbeispiel wird Eclipse in Version 3.5.1 [Eclipse] genommen. Zusätzlich wurde das ADT-Plugin (**A**ndroid **D**evelopment **T**ools) in der Version 0.9.3 [Alliance (a)] installiert. Als Programmiersprache kommt Java in der Version 1.6.0.16 [Sun Microsystems] zum Einsatz.
- Android SDK:
  Das Android SDK ist für Windows, Linux und Mac verfügbar. Es beinhaltet die API Dokumentation, Beispielprogramme und hilfreiche Tools

  - aapt (Verwaltung für Ressourcen-Dateien)
  - DexDump (Diagnosetool für DEX-Dateien[2])
  - dx (wandelt class-Dateien ins DEX Format um)
  - Android Emulator (Testen seiner Applikation)

  zur Unterstützung bei der Entwicklung [Voigt (2009)].
  In der Arbeit wird die Android SDK in der Version 1.6 [Alliance (a)] verwendet.

- Dalvik Debug Monitoring Service (DDMS):
  Der Dalvik Debug Monitoring Service (DDMS) ist eine Debugging Schnittstelle zwischen der IDE und der Applikation. Hier können Prozesse auf dem Emulator oder auf dem Android-Endgerät überwacht werden. Der DDMS bietet die Option den Sourcecode zu debuggen, wie beispielsweise gesetzte Haltepunkte oder den aktuellen Stand von Variablen zu überwachen. Zusätzlich bietet es Möglichkeiten, SMS und eingehende Anrufe zu simulieren, die entsprechenden Thread- und Heap Eigenschaften auszugeben oder den aktuellen Bildschirm aufzunehmen (Screenshot) (angelehnt an [Mosemann und Kose (2009)]).
- Android Asset Packaging Tool (AAPT):
  Mit dem Android Asset Packaging Tool (AAPT) lassen sich die apk-Dateien generieren. Sie enthalten sowohl die Binärdateien wie auch die Ressourcen (angelehnt an [Mosemann und Kose (2009)]).
- Android Debug Bridge (ADB):
  Mit Hilfe des Android Debug Bridge (ADB) kann der Emulator und das Android-Endgerät direkt von der Konsole aus gesteuert werden. Insbesondere lassen sich hier die apk-Dateien (Android Applikationen) auf beiden installieren (angelehnt an [Mosemann und Kose (2009)]).
- DroidDraw:
  GUI Designer für Android

## 2.5 Usability und Android

Im Rahmen dieser Arbeit wird das Thema Usability bearbeitet. In diesem Forschungsgebiet existieren Studien und Arbeiten, die sich mit Usability und mobilen Endgeräten beschäftigen. Der Artikel von Balagtas-Fernandez und Hussmann beschreibt das *EVAHELPER FRAMEWORK*, welche Informationen über Events und UI Komponenten in

[2] Class-Dateien werden für die Dalvik VM in DEX-Dateien umgewandelt.

einer Android Anwendung sammelt bzw. protokolliert. Die gesammelten Daten werden für eine einfache Usability-Bewertung mit Hilfe von Graph ML in einen Graphen überführt. In dem Framework müssen die Events und deren Komponenten manuell im Code hinzugefügt werden. Die manuelle Protokollierung sehen die Autoren als problematisch an, da sie für den Entwickler bei einem großen Projekt entsprechend viel Zeit benötigt. In einer Weiterentwicklung des Frameworks sollte die Protokollierung automatisiert werden. Ein möglicher Lösungweg wäre die Konfiguration der Protokollierung über eine grafische Oberfläche. In dieser könnten dann die Events und deren UI Komponenten in einer Anwendung für eine automatische Protokollierung ausgewählt werden. Eine weitere Möglichkeit besteht in der Protokollierung über die Aspektorientierte Programmierung (AOP).

Es existieren weitere Studien, die sich nicht speziell mit Android befassen, aber welche sich mit Usability im mobilen Bereich beschäftigen:

- **Mobile Anwendungen**
  In diesem Bereich gibt es Studien, bei denen die Usability für mobile Anwendungen untersucht worden ist. Ryan und Gonsalves haben in ihrer Studie zum Beispiel die Besonderheiten der Interaktion zwischen Benutzer und mobilen Endgerät beschrieben. Außerdem wurden hier Usability-Attribute zwischen einer Applikation auf dem Computer und einer mobilen Applikation untersucht. In der Studie von Karlson u. a. wurden im Labor verschiedene mobile Tasks in einer Laborstudie ausgeführt und die Usability-Attribute dabei genauer betrachtet. Die Studie von Keijzers u. a. untersucht ebenfalls Usability-Attribute auf Applikationen am PC und auf dem mobilen Endgerät.

- **Touch-Display**
  Die mobilen Endgeräte mit Android setzen vermehrt auf ein Touch-Display. Diese Interaktionsart stellt neue Usability-Anforderungen an das mobile Endgerät und deren Anwendungen. In den Artikeln von Park u. a. und Koskinen u. a. wurden diese diese beschrieben. Der Artikel von Gunawardana u. a. beschreibt außerdem noch die Problematik beim Schreiben mit dem Touch-Display.

- **Guidelines**
  Im Themengebiet Usability existieren eine Menge Guidelines, die die Usability bei mobilen Endgeräten beschreiben. In dem Artikel von Hussain und Ferneley werden auf diese verschiedenen Guidelines hingewiesen. Jene existieren für das IPhone [Inc. (2010)], für Android-Anwendungen [Alliance (2010)]. Außerdem gibt es Guidelines von Nokia (2010) und deren Community [Wiki (2010)].

Die Tabelle 2.1 vergleicht die unterschiedlichen Studien miteinander und ordnet sie ihren Themenbereichen zu. Dabei wurden die Guidelines außen vor gelassen, da sie nur Hinweise geben, wie man die UI aufzubauen und was dabei zu beachtet ist. Die folgenden Attribute für die Einteilung wurden verwendet:

- Benutzerinteraktion
  Die Benutzerinteraktion mit dem Touch-Display wurde auf neue Usability-Aspekte betrachtet.
- Usability-Attribute (s. Abschnitt 2.1.2)
  Die Usability-Attribute wurden in den Studien behandelt.
- PC / Mobile Applikation
  In den Studien wurde die Usability bei PC Applikationen und/oder Mobile Applikationen betrachtet.
- Benutzerbefragung
  Mit Hilfe von Benutzerbefragungen wurden subjektive Usability-Ergebnisse erfasst.
- Messdaten sammeln/auswerten
  Hier wurden relevante Usability-Daten anhand von Messgrößen gesammelt und/oder ausgewertet.
- Labor/Feld
  Die Usability-Untersuchungen wurden im Labor oder Feld durchgeführt.

| | Benutzer-interaktion | Usability-Attribute | PC Appl. | Mobile Appl. | Benutzer-. befragung | Messdaten sammeln/ auswerten | Labor | Feld |
|---|---|---|---|---|---|---|---|---|
| Balagtas-Fernandez und Hussmann (2009) | | X | | X | | X | X | |
| Ryan und Gonsalves (2005) | X | X | X | X | X | X | X | |
| Park u. a. (2008) | X | X | | X | | X | | |
| Koskinen u. a. (2008) | X | X | | X | X | X | X | |
| Gunawardana u. a. (2010) | X | X | | X | X | X | X | |
| Hussain und Ferneley (2008) | | X | | X | X | | | |
| Karlson u. a. (2010) | | X | | X | X | X | X | |
| Keijzers u. a. (2008) | | X | X | X | X | X | X | |

Tabelle 2.1: Vergleich der Studien

# 3 Analyse

In diesem Kapitel werden die Anwendungsfälle und Anforderungen an die Usability-Komponente mit Hilfe eines Anwendungsbeispiels herausgearbeitet. Dieses wird im folgenden Abschnitt 3.1 kurz vorgestellt und auf die Usability-Probleme des Anwendungsbeispiels eingegangen Im Kapitel 2.1.4 wurden Usability-Probleme bei mobilen Endgeräten beschrieben. Zusätzlich kommen noch die Usability- Probleme des Anwendungsbeispiels hinzu. Aus dem Anwendungsbeispiel und den Usability- Problemen werden in Abschnitt 3.2 die funktionalen Anforderungen, welche auch den Anwendungsfällen der Usability-Komponente entsprechen, erarbeitet und beschrieben. Außerdem werden noch die nicht-funktionalen Anforderungen der Usability-Komponente im Abschnitt 3.3 aufgeführt.

## 3.1 Anwendungsbeispiel

Verschiedene Studien [Mobilfunk (2009), Bundesamt (2009)] bestätigen eine starke Verbreitung mobiler Endgeräte in der Bevölkerung. In Kapitel 1 wurden bereits die steigenden Absatzzahlen mobiler Anwendungen erwähnt. Im Bereich der mobilen Anwendungen besteht also ein großes Potential. Sie werden in verschiedene Themenbereiche unterteilt: Ein Gebiet umfasst das E-Learning. Hiermit lassen sich Lernprozesse oder Lerneinheiten durch den Einsatz digitaler Technologien (zur Aufzeichnung, Speicherung, Übertragung, Be-/Verarbeitung, Anwendung und Präsentation von Informationen) umsetzen (angelehnt an [Wache]). Der Autor wird in dieser Arbeit nicht weiter auf das Thema E-Learning im Allgemeinen eingehen. Er verweist auf die etwaige Fachliteratur.

Für den User ist das mobile Endgerät fast immer verfügbar. Er hat unterwegs, zum Beispiel in der Bahn, Zeit zur Verfügung, die für E-Learning genutzt werden kann. Hierbei muss allerdings auf einige Merkmale der Entwicklung besonders geachtet werden:

- Hardware-Ressourcen, wie Bildschirmgröße oder Art der Benutzerinteraktion (Touchscreen; Tastatur) des mobilen Endgerätes
- kurze und kompakte Lerneinheiten
- schnelle Einarbeitung in die Lerneinheiten

Die Usability einer mobilen E-Learning Anwendung ist ein weiterer Faktor für den Erfolg. Schlechte Usability führt zu negativen Bewertungen im Portal. Dies kann das Aus für eine Anwendung bedeuten. In den mobilen E-Learning Anwendungen sollten daher die Usability-Probleme aufgespürt und beseitigt werden. Zusätzlich zu den Usability-Problemen aus Kapitel 2.1.4 können jene auftreten:

- Textlänge zu lang
  Der Bildschirm ist mit Text überfrachtet. Die Frage und die jeweiligen Antwortmöglichkeiten sind für den Bildschirm zu lang, um sie komplett anzuzeigen. Der User muss daher den Bildschirminhalt oft verschieben. Außerdem sind lange Texte nicht gut zu merken. Erschwerend kommen für den Benutzer unterwegs weitere Ablenkungen hinzu. Er müsste dadurch oft den Bildschirm verschieben, um die Frage noch einmal zu lesen. Dieses führt gleichzeitig zu einer Verlängerung der Ausführungszeit.

- Auswahlfelder/Buttons zu klein
  Ein zu kleines Auswahlfeld oder Button führt zu Problemen bei der Bedienung, da die Benutzer sie genau treffen müssen, um sie auszuwählen. Unterwegs kann es zu Rucklern kommen, wodurch die Auswahl erschwert wird. Dieses könnte dazu führen, dass der User öfter in der Anwendung navigieren oder die Felder öfter betätigen muss. Auch hierdurch wird die Ausführungszeit verlängert.

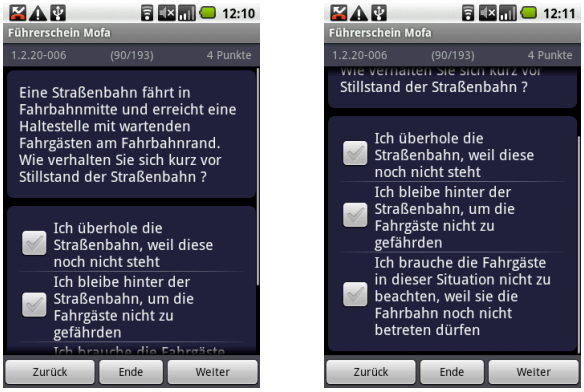
- Bedienfehler durch das Touch-Display
  Des Weiteren könnte der angezeigte Text mit Bildern für das Display zu groß sein, so dass er vom User verschoben werden muss. Hierbei können die Auswahlfelder für die Antworten ungewollt markiert werden.

- Speicherung von Lerninhalt bei Störungen
  Bei der Benutzung kann die E-Learning Anwendung durch SMS, Anruf oder auch durch den Benutzer selbst unterbrochen werden. Bei einer solchen Unterbrechung sollte die Anwendung den aktuellen Lernstatus der Lerneinheit speichern. Bei Wiederaufnahme sollte diese genau dort wieder fortgesetzt werden.

- Hilfefunktion
  Im Verlauf einer Lerneinheit kann es zu Problemen in der Bedienung kommen. Eine entsprechende Hilfefunktion kann hierbei dem Benutzer helfen. Ein zu häufiger Aufruf könnte aber darauf hinweisen, dass die Bedienung/Menüführung nicht eindeutig ist.

Anhand von den Applikationen „Führerschein Mofa", „Learn“ und „Fantastisch Memo“ werden die o.g. Probleme verdeutlicht.

In der Abbildung 3.1 wird ein Screenshot aus der Anwendung „Führerschein Mofa“ gezeigt. Die Länge des Textes führt zum Splitten des Bildschirminhaltes zwischen Frage und Antwort. Beim Lesen der Antworten könnte der Benutzer die Frage wieder vergessen haben, wenn er




(a) Frage (b) Antworten

Abbildung 3.1: Anwendung „Führerschein Mofa!“

zwischenzeitlich abgelenkt wurde. Daraufhin müsste er auf dem Bildschirm mehrmals auf- und abnavigieren.


Abbildung 3.2: Anwendung „Fantastisch Memo“

Die Abbildung 3.2 zeigt die Anwendung „Fantastisch Memo“. Die Buttons auf dem Bildschirm sind klein und nah beieinander. Daher ist die Auswahl der Buttons bei großen Fingern und bei Rucklern sehr schwierig. Die Auswahl kann dann jedoch nur umständlich wieder rückgängig gemacht werden.

Der Screenshot in der Abbildung 3.3 zeigt die Problematik des Auf- und Abnavigierens mit dem Touch-Display. Hierbei kann es unter anderem jedoch auch zur Auswahl einer Checkbox führen, obwohl dieses überhaupt nicht gewünscht war.

In den Screenshots aus Abbildung 3.4 ist zusätzlich zu der Frage ein Video eingeblendet. Bei einem ankommenden Gespräch wird das Video im Hintergrund unterbrochen und zeigt

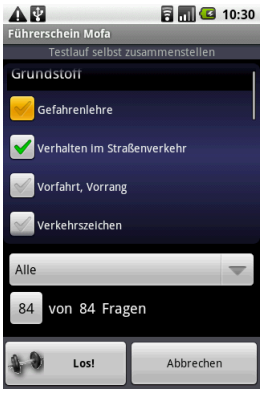
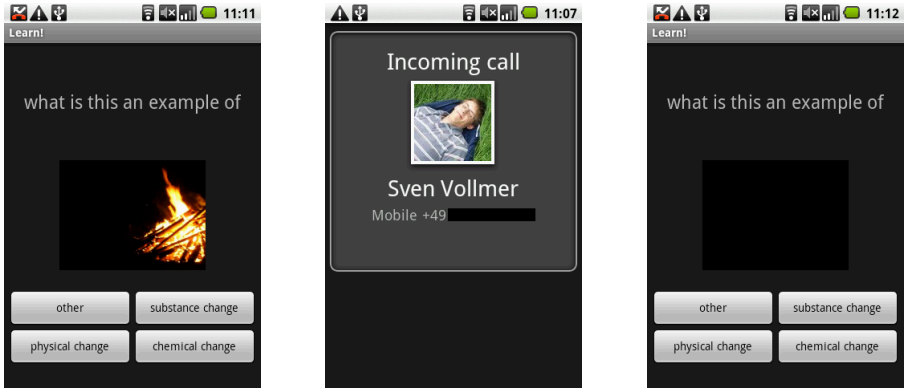



Abbildung 3.3: Anwendung “Führerschein“




(a) Lerneinheit bearbeiten

(b) Störung

(c) Fortsetzen der Lerneinheit nach Anruf

Abbildung 3.4: Anwendung „Learn!“

nur noch ein schwarzes Bild in dem Bereich an. Es ist jedoch kein Hinweis auf dem Bildschirm zu finden, um das Video erneut abzuspielen. Durch Ausprobieren des Autors wurde herausgefunden, dass ein erneutes Antippen das Video noch einmal startet.

## 3.2 Funktionale Anforderungen

In diesem Kapitel werden die funktionalen Anforderungen an die Usability-Komponente beschrieben.

### 3.2.1 Anwendungsfälle

Die Analyse des Anwendungsbeispiels zeigt wie wertvoll Usability für den Erfolg der mobilen Anwendung sein kann. Im vorherigen Abschnitt wurden die Usability-Probleme im mobilen Bereich erwähnt. Die zu entwickelnde Usability-Komponente sollte sich in eine bestehende Anwendung integrieren lassen, wo sie dann die Usability-Probleme aufzeichnen und später entsprechend auswerten. Daraus ergeben sich für die Entwicklung dieser Komponente folgende Anwendungsfall-Gruppen:

- Konfiguration
- Messen
- Protokollierung (Logging)
- Benutzerinteraktion
- Auswertung
- Kommunikation

Diese aufgeführten Gruppen werden in den folgenden Abschnitten im Detail beschrieben.

#### 3.2.1.1 Konfiguration

Die Komponente ist in die bestehende Anwendung zu integrieren, wofür sie entsprechend konfiguriert werden muss. Folgende Anwendungsfälle sind hierfür notwendig:

**Task definieren** In der Anwendung gibt es bestimmte Aufgaben, die der Benutzer ausführen kann. Der Usability-Experte legt fest, welche davon für eine Usability-Untersuchung relevant sind. (Anhang A.1.1)

**Task konfigurieren** Der Entwickler kann die Tasks vom Usability-Experten vorgegeben bekommen. Anschließend werden die entsprechenden Tasks konfiguriert. (Anhang A.1.2)

**Messgrößen festlegen** Um Rückschlüsse auf die Usability-Attribute (siehe Kapitel 2.1.2) zu erhalten, müssen die entsprechenden Messgrößen angegeben werden. Der Usability-Experte entscheidet, welche Messgrößen für die Anwendung sinnvoll sind, und legt die kritischen Werte für die spätere Auswertung fest. Diese Messgrößen werden im Abschnitt 3.2.1.2 genauer beschrieben. (Anhang A.1.3)

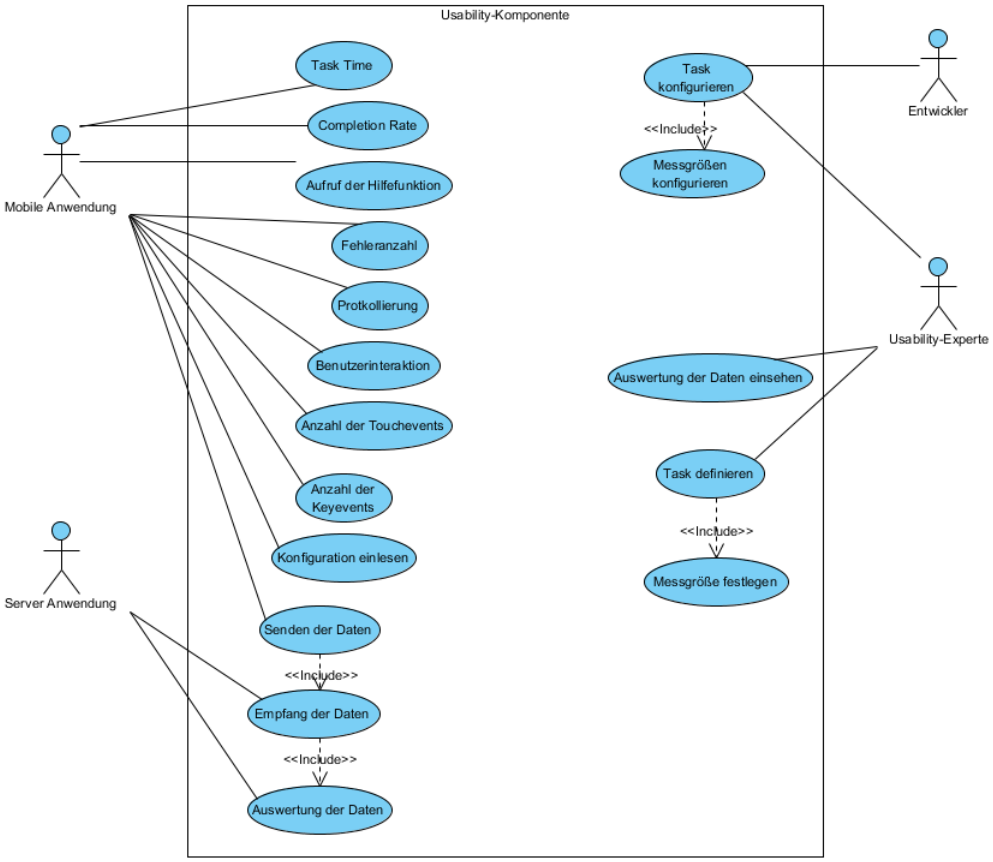



Abbildung 3.5: Usecase-Diagramm der Usability-Komponente

**Messgrößen konfigurieren** Die spezifischen Messgrößen müssen für die Tasks konfiguriert und an den entsprechenden Stellen im Sourcecode benutzt werden. Anhand der Dokumentation soll dem Entwickler Hilfestellung mit einfachen Beispielen gegeben werden. (Anhang A.1.4)

**Konfiguration einlesen** Die entsprechend angelegte Konfiguration muss für die Verarbeitung eingelesen werden. Nachdem Starten der mobilen Anwendung und vor dem Start der Messung sollte dieses geschehen. (Anhang A.1.5)

#### 3.2.1.2 Messen

Das Ermitteln der spezifischen Messgrößen gehört zu den wichtigsten Aufgaben der Usability-Komponente, denn sie weisen auf Usability-Probleme in der Anwendung hin. Folgendes ist in diese Anwendungsfall-Gruppe eingeordnet:

**Zeitaufwand für die Aufgabe (Task Time)** Der Benutzer kann wie oben beschrieben die Aufgaben (Tasks) ausführen. Hierfür benötigt er eine gewisse Zeit, die es zu messen gilt. Dadurch können entsprechend Rückschlüsse auf die Effektivität geschlossen werden. (Anhang A.2.1)

**Completion Rate** Die Effektivität beim Erfüllen der Aufgabe kann gemessen werden, wofür der Entwickler einen idealen Ablauf für das erfolgreiche Bearbeiten der Aufgabe bestimmt. Weicht der Benutzer von diesem idealen Pfad ab, ändert sich die Completion Rate entsprechend. (Anhang A.2.2)

**Fehleranzahl** Mit der Fehleranzahl sollen die Ausnahmen (Exceptions) in einem Programm gemessen werden. Mit deren Hilfe können sowohl Probleme bei der Anwendungsbedienung als auch bei der Programmierung aufgedeckt werden. (Anhang A.2.3)

**Aufruf der Hilfefunktion** In einer Anwendung werden Hilfefunktionen bei Bedienungsproblemen angeboten. Ein häufiger Aufruf kann auf eine unklare Menüführung/Anweisungen hinweisen. (Anhang A.2.4)

**Anzahl der Touch Events** Die Anzahl der so genannten unterschiedlichen Touch Events kann ebenso auf eine unklare Menüführung oder Probleme mit der Benutzeroberfläche hinweisen. (Anhang A.2.5)

**Anzahl der Key Events** Die Anwendung kann ebenfalls durch die Tasten bedient werden. Die Anzahl der so genannten unterschiedlichen Key Events kann auch auf eine unklare Menüführung oder Probleme mit der Benutzeroberfläche hinweisen. (Anhang A.2.6)

**Komplexität der Menüführung** Die Navigation durch die Benutzeroberflächen (Android: Activities) kann viele Zweige oder Unterzweige enthalten, die in einer Anwendung die Komplexität steigern und so eventuell zu Verwirrungen beim Benutzer führen können. (Anhang A.2.7)

#### 3.2.1.3 Protokollierung

Die Protokollierung kann den Entwickler dabei unterstützen, auf weitere Usability-Probleme hinzuweisen. Während der Laufzeit der Anwendung können die ausgeführten Aktionen des Benutzers aufgezeichnet werden, wie beispielsweise die quantitativen Ausprägungen der Messgrößen zu einem bestimmte Zeitpunkt. Die Protokollierung und quantitativen Ausprägungen werden für eine spätere Auswertung in einer Datenbank oder Datei gespeichert, welche beim Senden der Daten später auf den Server mitübertragen wird. (Anhang A.3)

#### 3.2.1.4 Benutzerinteraktion

Beim Starten der Anwendung muss der Benutzer gefragt werden, ob er an einer Usability-Untersuchung zur Verbesserung der Anwendung teilnehmen möchte. Der Hinweis muss Informationen über die Protokollierung, Datenerfassung und den Fragebogen enthalten. Wenn keine Zustimmung vom User erteilt wurde, liegt es in der Verantwortung des Entwicklers, die Usability-Untersuchung nicht zu starten. Zusätzlich kann ein Feedback-Fragebogen dem Benutzer zur Beantwortung vorgelegt werden. (Anhang A.4)

#### 3.2.1.5 Auswertung

Die quantitativen Ausprägungen der Messgrößen, die Protokollierung und der Feedback-Fragebogen werden während der Laufzeit aufgezeichnet. Im späteren Verlauf wird diese, nach Rückfrage an den Benutzer, zu dem Server gesendet; hierfür ist eine entsprechend höhere Bandbreite notwendig. Auf dem Server werden die Daten dann für den Usability-Experten aufbereitet.

**Auswertung der Daten** Die Daten werden von dem Benutzer an den Server geschickt. Dort werden die quantitativen Ausprägungen der Messgrößen und die Protokollierung für den Usability-Experten ausgewertet. (Anhang A.5.2)

**Auswertung der Daten einsehen** Der Usability-Experte kann die aufbereiteten Daten und die Protokolldateien seiner Anwendung einsehen. (Anhang A.5.1)

#### 3.2.1.6 Kommunikation

Für die Kommunikation zwischen mobilen Endgerät und dem Server sind die folgenden Anwendungsfälle notwendig:

**Senden der Daten** Die Daten für die Usability-Untersuchung und die Protokolldatei werden an den Server gesendet. (Anhang A.6.1)

**Empfang der Daten** Der Server empfängt die Daten von der mobilen Anwendung. (Anhang A.6.2)

## 3.3 Nicht-funktionale Anforderungen

Nachdem die funktionalen Anforderungen an die Usability-Komponente in Abschnitt 3.2 beschrieben worden sind, werden im folgenden Abschnitt die nicht-funktionalen Anforderungen erläutert. Diese sind in der Regel unabhängig von einem speziellen Anwendungsfall oder können aus anderen Gründen nicht als funktionale Anforderung in der Analyse beschrieben werden [Kahlbrandt (2001)]. Sie sind bei der Entwicklung der Komponente zu berücksichtigen und haben Auswirkungen auf den Entwurf der Software.

Die Komponente wird in bestehende Anwendungen integriert. Es gelten daher die allgemeinen Qualitätsmerkmale für Software-Produkte (siehe [Kahlbrandt (2001)]).

### 3.3.1 Effizienz

Durch die Integration der Usability-Komponente in die Anwendung kann die Verzögerungszeit vergrößert werden. Diese sollte bei maximal 500ms liegen. Die Usability-Komponente müsste daher mit den Hardware-Ressourcen des mobilen Gerätes sparsam umgehen. In späteren Tests der Anwendung, mit und ohne Usability-Komponente, sollte die Verzögerungszeit gegen den Wert gemessen werden. Hierbei liegt ein besonderes Augenmerk auf Protokollierung und Speicherung der Daten zur Laufzeit. Die beiden Anwendungsfall-Gruppen können die Anwendung verzögern. Es wird angenommen, dass sie zur Laufzeit die Hardware-Ressourcen am stärksten belasten.

### 3.3.2 Zuverlässigkeit (Robustheit)

Die zu entwickelnde Usability-Komponente sollte nicht unerwartet aufhören, zu funktionieren. Abstürze oder auch Fehler dürfen die Anwendung, welche die Usability-Komponente enthält, nicht unwiderruflich stoppen. Der User darf von den Fehlern nichts bemerken, da sie nicht direkt etwas mit der Anwendung zu tun haben.

### 3.3.3 Zuverlässigkeit (Korrektheit)

Bei Anwendungsabbruch durch den Benutzer oder Anwendungsabsturz dürfen keine ungültigen Messgrössen entstehen. Sie sind vor der Auswertung zu löschen.

### 3.3.4 Benutzbarkeit (Erlernbarkeit)

In der Arbeit wird eine Usability-Komponente für Anwendungen geschrieben. Diese wird in eine bestehende oder noch zu entwickelnde Anwendungen integriert. Die Integration und Konfiguration der Komponente sollte möglichst einfach gehalten werden. Eine gute Dokumentation mit Beispielen sorgt hierbei für einen besseren Einstieg für den Entwickler.

### 3.3.5 Sicherheit

Ein System ist sicher, wenn es unter den vorgegebenen Bedingungen keine unzulässigen Ereignisse zulässt. Vor allem soll es zur Laufzeit nicht in unzulässige Zustände wechseln (angelehnt an [Kahlbrandt (2001)]). Dabei wird zwischen Security und Safety unterschieden (übersetzt heißen beide im deutschen Sicherheit [dict.cc]).

#### 3.3.5.1 Safety

Safety bezeichnet die Datenintegrität, die auch beim Fehlverhalten der Usability-Komponente gewährleistet werden sein muss. Sie soll sicherstellen, dass beim Speichern und Transport die Daten nicht beschädigt oder verändert werden, denn dieses kann die spätere Auswertung verfälschen.

#### 3.3.5.2 Security

Security bezeichnet den Schutz vor unbefugter Nutzung und anderem Missbrauch. In der Usability-Komponente werden die Daten auf dem mobilen Endgerät und später auch auf dem Server für die Verarbeitung gespeichert. Es handelt sich zwar hierbei nicht um sensible Daten, aber ein unbefugter Zugriff und gleichzeitige Veränderung der Daten (Safety) kann negativen Einfluss auf die spätere Auswertung haben. Aus diesem Grund sollten die Daten geschützt werden. Außerdem ist der spätere Usability-Server über das Internet erreichbar und muss deshalb vor dem unbefugten Zugriff geschützt werden.

### 3.3.6 Wartbarkeit (Verständlichkeit)

Die Integration und das Verwenden der Usability-Komponente in einer Anwendung sollte dem Entwickler erleichtert werden. Dies kann durch aussagekräftige Methodennamen und durch eine gute Dokumentation erreicht werden, die ebenfalls für eine zukünftige Weiterentwicklickung notwendig sind.

## 3.4 Zusammenfassung

In diesem Kapitel wurde ein Beispiel für mobile Anwendungen auf ihre Usability-Probleme hin analysiert. Für das Messen von Usability soll eine Usability-Komponente entwickelt werden, welche sich in eine bestehende Anwendung integrieren lässt. Daraufhin wurden die funktionalen Anforderungen, welche den Anwendungsbeispielen entsprechen, für eine solche Komponente aufgestellt. Mit Hilfe des Anwendungsbeispiels und der Usability-Probleme wurden Anwendungsfall-Gruppen (s. Abschnitt 3.2.1) für diese herausgearbeitet. In der Tabelle 3.1 sind diese Usability-Probleme aus dem Kapitel 2.1.4 und dem Anwendungsbeispiel aus Abschnitt 3.1 aufgelistet, wie auch deren Bezug zu den funktionalen Anforderungen. Die nicht-funktionalen Anforderungen in Abschnitt 3.3 spiegeln die Qualitätsmerkmale der Komponente wider. Diese müssen für den Entwurf der Komponente berücksichtigt werden. Im weiteren Verlauf der Arbeit wird im folgenden Kapitel 4 ein Architekturentwurf einer solchen Usability-Komponente vorgestellt. Als konkrete Ausprägung dieser Architektur wird ein Prototyp der Usability-Komponente entwickelt, der zur Verdeutlichung der Bestandteile der Architektur dient.

| Usability-Problem | Vorkommen | Anwendungsfall |
|---|---|---|
| Auswahlfelder/Buttons zu klein | Anw. | Task Time (s. Abschnitt 3.2.1.2), Touch Events (s. Abschnitt 3.2.1.2) |
| Textlänge zu lang | Anw. | Task Time (s. Abschnitt 3.2.1.2), Touch Events (s. Abschnitt 3.2.1.2) |
| Bedienfehler durch Touchdisplay | Anw. | TaskTime (s. Abschnitt 3.2.1.2), Touch Events (s. Abschnitt 3.2.1.2) |
| Lerneinheit bei Störung speichern | Anw. | - |
| Hilfefunktion | Anw.. | Aufruf der Hilfefunktion (s. Abschnitt 3.2.1.2) |
| Auftreten von Fehlern | Allg. | Fehleranzahl (s. Abschnitt 3.2.1.2) |
| Probleme beim Lösen von Aufgaben | Allg. | Completion Rate (s. Abschnitt 3.2.1.2) |
| Bedienfehler | Allg. | TaskTime (s. Abschnitt 3.2.1.2), Touch Events (s. Abschnitt 3.2.1.2) |

Tabelle 3.1: Vergleich der Usability-Probleme mit den Anwendungsfällen

# 4 Design

Das folgenden Kapitel beschreibt den Architekturentwurf der Usability-Komponente. Der Abschnitt 4.1 zeigt die vorhandene Hardware- und Softwareumgebung von den mobilen Endgeräten und dem Server. In Abschnitt 4.2 werden die einzelnen Schnittstellen der Komponenten innerhalb der Usability-Komponente beschrieben, insbesondere deren Einsatz auf dem Server und dem mobilen Endgerät. Im Abschnitt 4.3 wird der Feinentwurf der Komponenten und ihr Zusammenspiel dargestellt.

## 4.1 Systemarchitektur

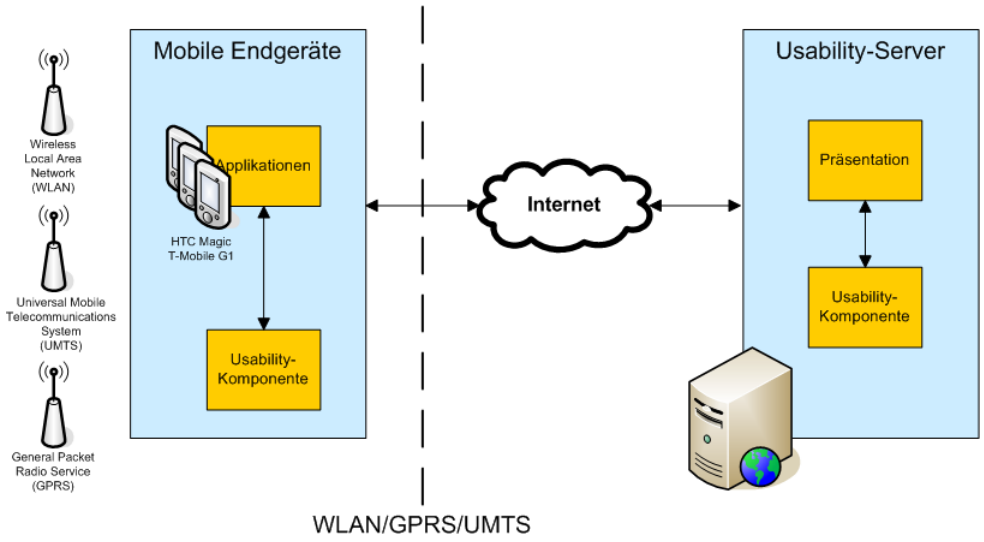



Abbildung 4.1: Erste Version der Systemarchitektur

In Abb. 4.1 wird der Aufbau der Systemarchitektur verdeutlicht. Auf der einen Seite befinden sich hierbei die mobilen Endgeräte, auf denen die Applikationen laufen. In Zusammenarbeit mit der Usability-Komponente wird Usability gemessen. Das Senden der Ergebnisse an den Server erfolgt über eine Datenverbindungen WLAN/GPRS/UMTS. Auf der anderen Seite werden die Ergebnisse vom Server entgegengenommen. Dort werden die verschiedenen

Resultate der mobilen Endgeräte ausgewertet. Die Präsentationsschicht bereitet die Daten entsprechend auf.

In der ersten Version der Systemarchitektur (s. Abb. 4.1) wird eine Internetverbindung benötigt, um die zukünftigen Daten der Usability-Untersuchung über diese Verbindung vom mobilen Endgerät zum Usability-Server zu übertragen. Als Übertragungsarten für die Daten kommen z.B. WLAN, GPRS oder UMTS in Frage; für jede davon können aber unterschiedliche Kosten entstehen. Diese differenzierte Betrachtung fehlt in der ersten Systemarchitektur. Ein weiterer wichtiger Aspekt in der zweiten Version liegt in der Anzahl der mobilen Endgeräte pro Usability-Server. Ein solcher Server kann nur eine begrenzte Anzahl von Verbindungen zulassen. Außerdem spielt die Anzahl der Anwendungen pro Usability-Server eine wichtige Rolle, insbesondere für eine spätere Lastverteilung und den Administrationsaufwand.

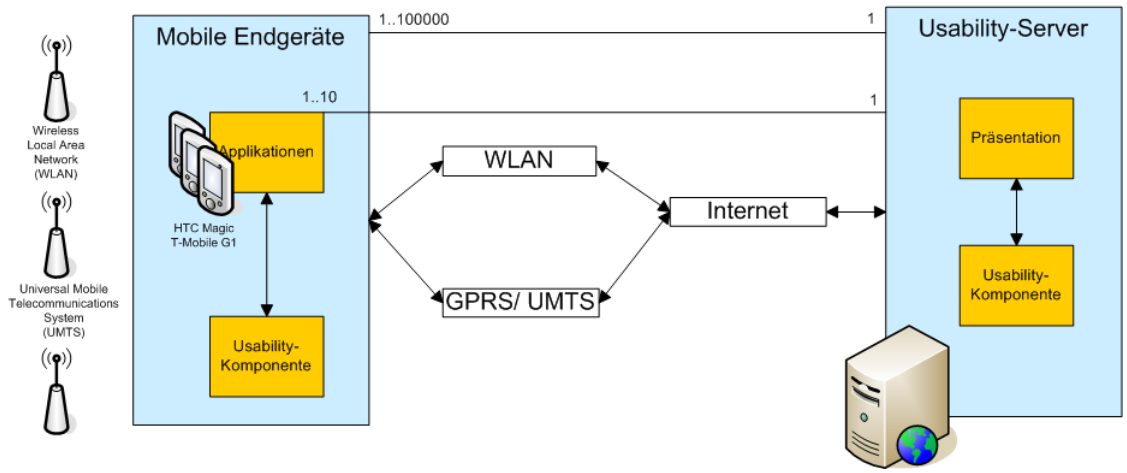



Abbildung 4.2: Zweite Version der Systemarchitektur

Die Abbildung 4.2 zeigt die zweite Version der Systemarchitektur. Die Übertragungsarten werden hier getrennt betrachtet. Bei großen Datenmengen sollte WLAN benutzt werden, da es sich dort um eine kostengünstigere Übertragungsart handelt bzw. dort eher nicht per Datenvolumen abgerechnet wird. Kleinere Datenmengen können über das kostenpflichtige Übertragungsprotokoll GPRS/UMTS verschickt werden. Ein entsprechender Hinweis sollte dem Benutzer auf dem mobilen Endgerät gegeben werden, welche Art der Datenübertragung genutzt wird. Zusätzlich unterscheidet die Systemarchitektur die Anzahl der mobilen Endgeräte und Anwendungen pro Usability-Server. Die Zahl der mobilen Endgeräte wird auf bis zu 100000 festgelegt. Die Anzahl der Applikationen pro Server liegt bei 10. Diese Zahlen sollte in späteren Tests überprüft werden. Dies ist aber nicht Gegenstand der aktuellen Arbeit.

### 4.1.1 Mobile Endgeräte

Dem Autor stehen zwei verschiedene mobile Endgeräte zur Verfügung, die jeweils von der Firma HTC [htc.com (2009)] kommen. Der Autor konzentriert sich dabei nur auf die wichtigen Details der mobilen Endgeräte. Zusätzlich bietet das SDK von Android einen Emulator an. Mit dessen Hilfe lassen sich unterschiedliche mobile Endgräte oder auch die verschiedenen Betriebssystem-Versionen von Android nachbilden. Sowohl auf den mobilen Endgeräten als auch auf dem Emulator wird die Applikation mit der integrierten Usability-Komponente ausgeführt.

(a) HTC Magic (b) T-Mobile G1

Abbildung 4.3: Mobile Endgeräte

Das T-Mobile G1 (s. Abb. 4.3b) ist das erste mobile Endgerät, welches mit dem Android-Betriebsystem ausgeliefert wurde. Das HTC Magic (s. Abb. 4.3a) dagegen ist das Google-Handy der zweiten Generation. In der Hardware unterscheiden sich die beiden Geräte nur in einem Punkt. Das T-Mobile G1 besitzt im Gegensatz zum HTC Magic eine QWERTY Tastatur. Beide Smartphones laufen mit Android in der Version 1.6.

### 4.1.2 Usability-Server

**Hardware** Es wird ein Rechner mit Internetanschluss benötigt. Idealerweise sollte der Rechner eine hohe Verfügbarkeit (24/7)[1] besitzen.

#### Software

- Apache Webserver

[1] bezeichnet hier die ständige Verfügbarkeit des Dienstes und zwar 24 Stunden am Tag und 7 Tage die Woche

- Applikationserver (JBoss)
- Java
- Serverapplikation mit integrierter Usability-Komponente und einer Präsentationsschicht

## 4.2 Anwendungsarchitektur

Bernd Oesterreich [Oestereich (2006), S.158] nennt das fachliche Subsystemmodell als Möglichkeit, Anwendungsarchitekturen zu beschreiben. In der Anwendungsarchitektur dieser Arbeit wird das System in einzelne fachliche Untereinheiten (Komponenten) zerlegt, die im weiteren Verlauf genauer beschrieben werden wie ebenso ihr Zusammenspiel.

### 4.2.1 Usability-Komponente

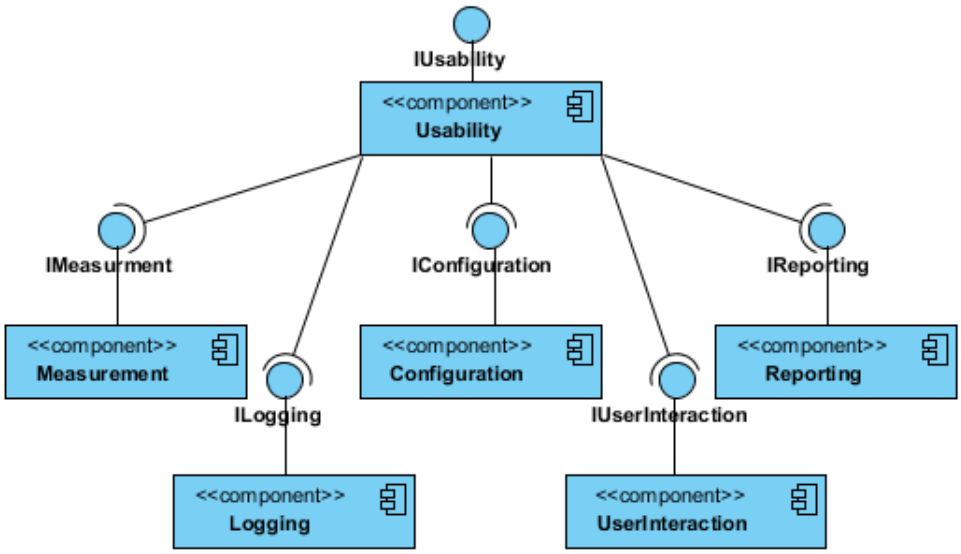



Abbildung 4.4: Fachliches Komponentenmodell

Die Anwendungarchitektur der Usability-Komponente besteht aus einem fachlichen und technischen Komponentenmodell. Um die gestellten Anforderungen (s. Kapitel 3.2 und 3.3) erfüllen zu können, beinhaltet die Darstellung (s. Abb. 4.4) die fachlichen Komponenten *Measurement*, *Configuration*, *Reporting*, *Logging* und *UserInteraction*. Zusätzlich kommen die Komponenten *Persistence*, *ClientCommunication* und *ServerCommunication* aus dem technischen Komponentenmodell (s. Abb. 4.5) hinzu. Um den Zugriff später transparenter zu

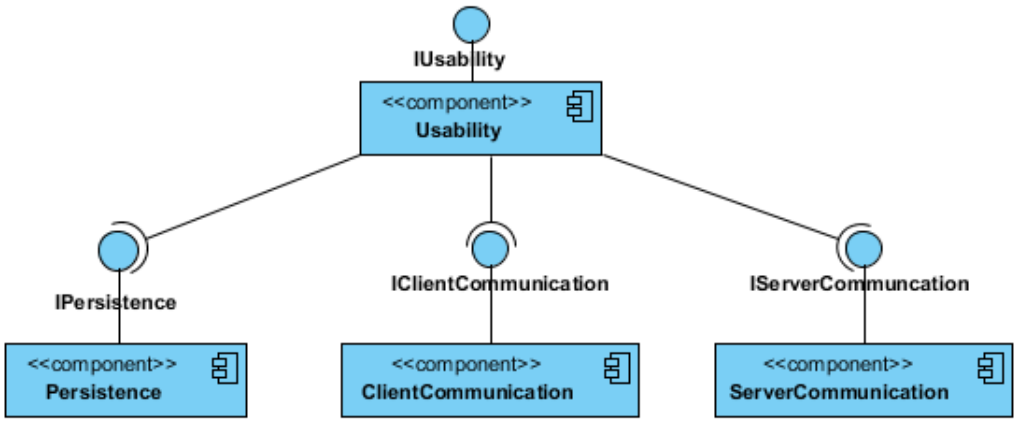



Abbildung 4.5: Technisches Komponentenmodell

machen, werden die Komponenten nach außen hin verborgen. Hierfür wird die zusätzliche Komponente *Usability* benötigt. Diese bietet eine Schnittstelle nach außen an und delegiert die Aufrufe an die entsprechenden Komponenten. Die aufgezählten Schnittstellen repräsentieren die Design-Anforderungen für die Usability-Komponente.

#### 4.2.1.1 Persistenz

Die Usability-Komponente erstellt während der Benutzung Messergebnisse (s. Abschnitt 4.2.1.5) und Feedback-Daten (s. Abschnitt 4.2.1.7). Diese Daten müssen auf dem mobilen Endgerät zwischengespeichert werden, um sie später an den Server zu senden; diese Aufgabe übernimmt die Komponente *Persistence* (s. Abb. 4.6). Die folgenden Abschnitte beschreiben die Funktionen der Schnittstelle IPersistence.

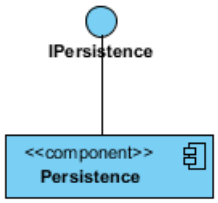



Abbildung 4.6: Komponente *Persistence*

**Messergebnisse lesen** Die Messergebnisse werden über die Schnittstelle eingelesen. Dabei werden sie aus einer externen sequenziellen Darstellungsform in Objekte deserialisiert.

**Messergebnisse speichern** Die Messergebnisse werden über die Schnittstelle auf dem mobilen Endgerät gespeichert. Sie liegen dabei als Objekte vor und werden in eine sequenzielle Darstellungsform serialisiert.

**Feedback-Daten lesen** Die Feedback-Daten werden mit dieser Methode eingelesen. Sie werden aus einer externen sequenziellen Darstellungsform in Objekte deserialisiert.

**Feedback-Daten speichern** Die Schnittstelle ermöglicht die Speicherung der Feedback-Daten auf dem mobilen Endgerät. Die Feedback-Daten liegen dabei als Objekte vor und werden in eine externe sequenzielle Darstellungsform serialisiert.

#### 4.2.1.2 Client-Kommunikation

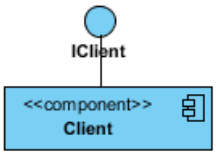



Abbildung 4.7: Komponente *ClientCommunication*

Die Komponente *ClientCommunication* (s. Abb. 4.7) ist für die Kommunikation des mobilen Endgerätes (s. Abschnitt 3.2.1.6) zuständig. Die Methoden der Schnittstelle IClientCommunication werden in den nachfolgenden Abschnitten beschrieben.

**Senden der Daten** Mit Hilfe der Methode kann eine Liste von Dateien an den Usability-Server gesendet werden. Hierbei wird ein Kommunikationskanal zum Server aufgebaut, und über den Kanal die Dateien gesendet. Diese Funktion erfüllt die Anforderung aus Kapitel 3.2.1.6.

#### 4.2.1.3 Server-Kommunikation

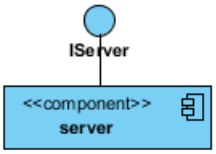



Abbildung 4.8: Komponente *ServerCommunication*

Die Komponente *ServerCommunication* (s. Abb. 4.8) ist für die Kommunikation zwischen dem Server und dem mobilen Endgerät zuständig (s. Abschnitt 3.2.1.6). Die Funktionalitäten der Schnittstelle IServerCommunication werden in den folgenden Abschnitten erläutert.

**Server starten** Die Methode startet den Usability-Server und horcht an einem Port auf Verbindungsanfragen von mobilen Endgeräten. Die Anforderung stammt aus Kapitel 3.2.1.6.

#### 4.2.1.4 Auswertung

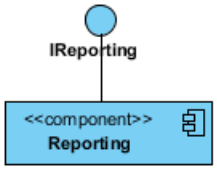



Abbildung 4.9: Komponente *Reporting*

Die Komponente *Reporting* (s. Abb. 4.9) wertet die auf dem Server befindlichen Usability-Ergebnisse aus (s. Abschnitt 3.2.1.5). Die nachfolgenden Abschnitte zeigen die Funktionen der Schnittstelle IReporting.

**Messergebnisse einlesen** Für die Auswertung müssen die gespeicherten Messergebnisse auf dem Server eingelesen werden. Hierbei werden sie aus einer sequenziellen Darstellungsform in Objekte deserialisiert. Die Methode erfüllt ein Teil der Anforderung aus Kapitel 3.2.1.5

**Auswertung erstellen** Mit dieser Funktion wird eine Auswertung der Messergebnisse vorgenommen. Auch sie ist ein Bestandteil der Anforderung aus Kapitel 3.2.1.5

#### 4.2.1.5 Messen

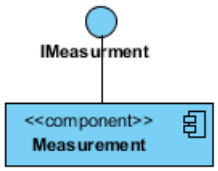



Abbildung 4.10: Komponente *Measurement*

Die Kernkomponente des Systems wird durch die Komponente *Measurement* (s. Abb. 4.10) repräsentiert, die für die Messung von Usability in der Anwendung (s. Abschnitt 3.2.1.2) zuständig ist. Die Funktionen der Schnittstelle IMeasurement werden in den folgenden Abschnitten dargestellt.

**Messung starten** Die Methode startet die Messung für den übergebenen Task.

**Messung stoppen** Die Messung für einen Task wird gestoppt.

**Messung erstellen** Die Methode erstellt die Messung für einen Task.

**Fehler hinzufügen** Hierüber werden die Fehler, die in einer Anwendung auftreten, hinzugefügt. Die Funktion ist ein Bestandteil der Anforderung aus Kapitel 3.2.1.2

**Task abrufen** Über den Namen wird der Task abgerufen.

**Exception hinzufügen** Die Methode kann Exceptions hinzufügen. Sie ist ebenso ein Bestandteil der Anforderung aus Kapitel 3.2.1.2.

**Aufruf einer Hilfefunktion hinzufügen** Hiermit werden Aufrufe der Hilfefunktion in der Komponente *Measurement* vermerkt. Diese Funktion bildet einen Teil der Anforderung aus Abschnitt 3.2.1.2 ab.

**Perfekten Pfad für einen Task angeben** Für die Anforderung aus Kapitel 3.2.1.2 ist diese Methode notwendig. Sie gibt der perfekten Pfad für einen Task an.

**Pfad hinzufügen** Hierbei wird ein Pfad-Element übernommen. Die Funktion wird ebenso benötigt, um der Anforderung aus Kapitel 3.2.1.2 nachzukommen.

**Menü-Element hinzufügen** Aufgerufene Menü-Elemente können über diese Methode hinzugefügt werden. Diese Anforderung stammt aus Kapitel 3.2.1.2.

**Key Events hinzufügen** Key Events werden hierüber hinzugefügt. Sie wird der Anforderung aus Kapitel 3.2.1.2 gerecht.

**Touch Event hinzufügen** Das Hinzufügen von Touch Events erfolgt durch diese Methode, die die Anforderung aus Kapitel 3.2.1.2 erfüllt.

**Pausieren der Anwendung** Die Zustandsänderung der Anwendung mit *onPause* hat Einfluss auf die Messgröße. Diese Funktion passt die Messgröße entsprechend an.

**Wiederaufnehmen der Anwendung** Eine weitere Zustandsänderung wird durch *onResume* ausgelöst. Diese Methode passt ebenfalls die Messgröße entsprechend an.

**Beenden der Anwendung** Die letzte Zustandsänderung wird durch *onDestroy* ausgelöst. Eine entsprechende Anpassung der Messgrößen wird hiermit durchgeführt.

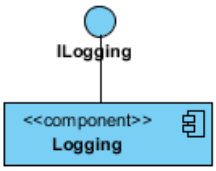



Abbildung 4.11: Komponente *Logging*

#### 4.2.1.6 Protokollierung

Die Komponente *Logging* (s.Abb. 4.11) ist für das Protokollieren (s. Abschnitt 3.2.1.3) zuständig. Die folgenden Abschnitte erläutern die Funktionen der Schnittstelle ILogging.

**Protokollierung schreiben** Das Schreiben von Ausgaben in eine Protokolldatei erfolgt über diese Methode. Sie erfüllt die Anforderung aus Kapitel 3.2.1.3.

**Protokolldatei abfragen** Die aktuelle Protokolldatei kann hierüber abgefragt werden.

#### 4.2.1.7 Benutzer-Interaktion

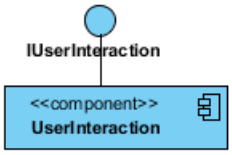



Abbildung 4.12: Komponente *UserInteraction*

Die Komponente *UserInteraction* (s. Abb. 4.12) ist für die Benutzer-Interaktion (s. Abschnitt 3.2.1.4) zuständig. Die nachfolgenden Abschnitte beschreiben die Aufgaben der Schnittstelle IUserInteraction.

**Abfrage-Dialog über die Teilnahme laden** Hiermit wird ein Dialog gestartet, ob der Benutzer an der Usability-Untersuchung teilnehmen möchte. Diese Funktion ist Bestandteil der Anforderung aus Kapitel 3.2.1.4.

**Zustimmung abfragen** Die Zustimmung des Benutzers kann über die Schnittstelle abgefragt werden.

**Konfigurierbaren Feedback-Fragebogen laden** Die Methode erlaubt, einen Feedback-Fragebogen mit eigenen Fragen und Antworten aufzurufen. Sie ist ebenso Bestandteil der Anforderung aus Kapitel 3.2.1.4.

**Ist der Feedback-Fragebogen komplett ausgefüllt?** Die Abfrage, ob der Fragebogen komplett ausgefüllt wurde, wird über diese Methode gelesen.

**Feedback-Ergebnis abfragen** Das Ergebnis der Befragung kann hiermit abgefragt werden.

#### 4.2.1.8 Konfiguration

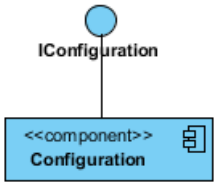



Abbildung 4.13: Komponente *Configuration*

Die Komponente *Configuration* (s. Abb. 4.13) ist für die Konfiguration (s. Abschnitt 3.2.1.1) zuständig, bei der die Tasks und auszuführenden Messmethoden für die Applikation angegeben werden. Die Funktionalität der Schnittstelle IConfiguration wird im folgenden Abschnitt beschrieben.

**Einlesen der Konfiguration** Die Methode liest die Konfiguration ein. Die Usability-Komponente wird auf diese Weise entsprechend angepasst. Sie erfüllt die Anforderung aus Kapitel 3.2.1.1.

#### 4.2.1.9 Usability

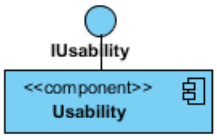



Abbildung 4.14: Komponente *Usability*

Die Komponente *Usability* verbirgt die oben genannten Komponenten und bietet nur eine Schnittstelle nach außen an. In den folgenden Abschnitten werden die Funktionen der Schnittstelle IUsability beschrieben. Auf die Speicherung der Daten hat der Anwendungsentwickler für das mobile Endgerät keinen Einfluss. Die Zusammenarbeit mit der Komponente *Persistence* wird in die entsprechenden Funktionen eingebunden.

**Daten an Server senden** Über den Pfad der mobilen Anwendung werden die zu sendenden Dateien ausgelesen und mit Hilfe der Komponente *ClientCommunication* (s. Abschnitt 4.2.1.2) und deren Funktion *Senden der Daten* (s. Abschnitt 4.2.1.2) gesendet.

**Server starten** Mit Hilfe des Serverpfades und dem Port wird der Server über die Komponente *ServerCommunication* (s. Abschnitt 4.2.1.3) und deren Methode *Server starten* (s. Abschnitt 4.2.1.3) initialisiert.

**Starte Auswertung** Der Serverpfad muss beim Starten der Auswertung übergeben werden. Die Komponente *Reporting* (s. Abschnitt 4.2.1.4) ist hierbei beteiligt. Für die Auswertung werden die folgenden Funktionen der Komponente verwendet:

1. *Messergebnisse einlesen* (s. Abschnitt 4.2.1.4)
2. *Messergebnisse lesen* (s. Abschnitt 4.2.1.1)
3. *Feedback-Daten lesen* (s. Abschnitt 4.2.1.1)
4. *Erstellen der Auswertung* (s. Abschnitt 4.2.1.4)

**Messung starten** Über den Tasknamen wird die Messung mit der Methode *Messung starten* (s. Abschnitt 4.2.1.5) in der Komponente *Measurement* (s. Abschnitt 4.2.1.5) initialisiert.

**Messung stoppen** Die Messung für einen Task wird mit der Methode *Messung stoppen* (s. Abschnitt 4.2.1.5) aus der Komponente (s. Abschnitt 4.2.1.5) gestoppt. Danach werden die Messgrößen zu dem Task mit der Funktion *Messergebnisse speichern* (s. Abschnitt 4.2.1.1) in der Komponente *Persistence* (s. Abschnitt 4.2.1.1) gespeichert.

**Fehler hinzufügen** Für einen Task können Fehler hinzugefügt werden. Dabei wird die Methode *Fehler hinzufügen* (s. Abschnitt 4.2.1.5) der Komponente *Measurement* (s. Abschnitt 4.2.1.5) verwendet.

**Exception hinzufügen** Außerdem kann für einen Task die Anzahl der Exceptions gezählt werden; dieses übernimmt die Komponente *Measurement* (s. Abschnitt 4.2.1.5). Die Methode *Exception hinzufügen* (s. Abschnitt 4.2.1.5) wird hierbei verwendet.

**Aufruf einer Hilfefunktion hinzufügen** Über die Funktion *Aufruf der Hilfefunktion hinzufügen* (s. Abschnitt 4.2.1.5) der Komponente *Measurement* (s. Abschnitt 4.2.1.5) werden die Aufrufe der Hilfefunktion gezählt.

**Perfekten Pfad für einen Task angeben** Der perfekte Pfad wird über die Methode *Perfekten Pfad für einen Task angeben* (s. Abschnitt 4.2.1.5) der Komponente *Measurement* (s. Abschnitt 4.2.1.5) angegeben.

**Pfad hinzufügen** Das Hinzufügen des Pfades erfolgt über die Funktion *Pfad hinzufügen* (s. Abschnitt 4.2.1.5) der Komponente *Measurement* (s. Abschnitt 4.2.1.5).

**Menü-Element hinzufügen** Die Methode *Menü-Element hinzufügen* (s. Abschnitt 4.2.1.5) in der Komponente *Measurement* (s. Abschnitt 4.2.1.5) fügt die Menü-Elemente ein.

**Key Event hinzufügen** Das Hinzufügen von Key Events übernimmt die Komponente *Measurement* (s. Abschnitt 4.2.1.5) und deren Funktion *Key Event hinzufügen* (s. Abschnitt 4.2.1.5).

**Touch Event hinzufügen** Das Hinzufügen von Touch Events übernimmt die Methode *Touch Event hinzufügen* (s. Abschnitt 4.2.1.5) in der Komponente *Measurement* (s. Abschnitt 4.2.1.5).

**Pausieren der Anwendung** Die Messgrößen für alle Tasks werden über die Komponente *Measurement* (s. Abschnitt 4.2.1.5) mit Hilfe der Funktion *Pausieren der Anwendung* (s. Abschnitt 4.2.1.5) angepasst.

**Wiederaufnehmen der Anwendung** Die Messgrößen für alle Tasks werden über die Komponente *Measurement* (s. Abschnitt 4.2.1.5) mit Hilfe der Methode *Wiederaufnehmen der Anwendung* (s. Abschnitt 4.2.1.5) angepasst.

**Beenden der Anwendung** Die Messgrößen für alle Tasks werden über die Komponente *Measurement* (s. Abschnitt 4.2.1.5) mit Hilfe der Funktion *Beenden der Anwendung* (s. Abschnitt 4.2.1.5) angepasst.

**Protokollierung schreiben** Die Protokollierung wird über die Komponente *Logging* (s. Abschnitt 4.2.1.6) und deren Methode *Protokollierung schreiben* (s. Abschnitt 4.2.1.6) durchgeführt.

**Protokolldatei abfragen** Die Protokolldatei kann über Komponente *Logging* (s. Abschnitt 4.2.1.6) und dessen Funktion *Protokolldatei abfragen* (s. Abschnitt 4.2.1.6) abgefragt werden.

**Abfrage-Dialog über die Teilnahme laden** Der Zugriff findet über die Komponente *UserInteraction* (s. Abschnitt 4.2.1.7) statt. Dabei wird die Methode *Abfrage-Dialog über die Teilnahme laden* (s. Abschnitt 4.2.1.7) der Komponente benutzt.

**Zustimmung abfragen** Über die Komponente *UserInteraction* (s. Abschnitt 4.2.1.7) wird mit Hilfe der Funktion *Zustimmung abfragen* (s. Abschnitt 4.2.1.7) das Ergebnis über die Zustimmung abgefragt.

**Konfigurierbaren Feedback-Fragebogen laden** Auch hier wird auf die Komponente *UserInteraction* (s. Abschnitt 4.2.1.7) über die Methode *Konfigurierbaren Feedback-Fragebogen laden* (s. Abschnitt 4.2.1.7) zugegriffen.

**Ist der Feedback-Fragebogen komplett ausgefüllt?** Die Abfrage wird über die Funktion *Ist der Feedback-Fragebogen komplett ausgefüllt?* der Komponente *UserInteraction* (s. Abschnitt 4.2.1.7) ausgeführt.

**Konfiguration lesen** Die Konfiguration wird über die Komponente *Configuration* (s. Abschnitt 4.2.1.8) mit Hilfe der Methode *Einlesen der Konfiguration* (s. Abschnitt 4.2.1.8) gelesen.

### 4.2.2 Mobiles Endgerät

Die Abb. 4.15 zeigt das Komponentenmodell für das mobile Endgerät, für das nicht alle Funktionen der Komponente *Usability* benötigt werden. Mit Hilfe der Komponenten *Client* werden die entsprechenden Schnittstellen für das mobile Endgerät gekapselt.

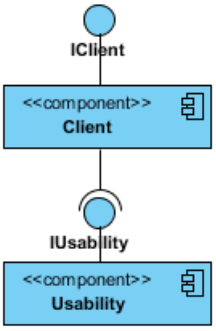



Abbildung 4.15: Komponentenmodell vom mobilen Endgerät

#### 4.2.2.1 Client

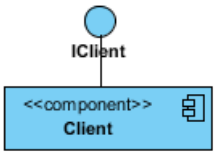



Abbildung 4.16: Komponente *Client*

Die Schnittstelle IClient benutzt die Funktionen der Komponente *Usability* (s. Abschnitt 4.2.1.9) und deren Schnittstelle IUsability, die für das mobile Endgerät von Bedeutung sind. Es werden alle Methoden bis auf *Server starten* und *Starte Auswertung* verwendet.

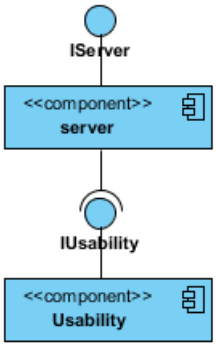



Abbildung 4.17: Komponentenmodell vom Server

### 4.2.3 Usability-Server

Die Abb. 4.17 zeigt das Komponentenmodell für den Server. Auch hier werden nicht alle Funktionen der Komponente *Usability* benötigt. Die Kapselung zur Komponente *Usability* übernimmt die Komponente *Server*.

#### 4.2.3.1 Server

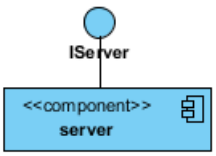



Abbildung 4.18: Komponente *Server*

Die Schnittstelle IServer benutzt die Funktionen der Komponente *Usability* (s. Abschnitt 4.2.1.9) und deren Schnittstelle IUsability, die für den Usability-Server relevant sind. Dazu gehören die Funktionen *Server starten* (s. Abschnitt 4.2.1.9) und *Starte Auswertung* (s. Abschnitt 4.2.1.9).

## 4.3 Feinentwurf

Beim Feinentwurf werden die Komponenten und ihre bereitgestellten Schnittstellen aus der Anwendungsarchitektur in Abschnitt 4.2 weiter verfeinert und ergänzt. Zur Darstellung der Komponenten werden Klassendiagramme verwendet. Zusätzlich wird in den Komponenten auf die wichtigen Implementierungsdetails hingewiesen. Für das mobile Endgerät und den

Usability-Server werden außerdem die Interaktion der Komponenten mit Hilfe von Sequenzdiagrammen aufgezeigt.

### 4.3.1 Usability-Komponente

In diesem Abschnitt werden die Komponente innerhalb der Usability-Komponente beschrieben.

#### 4.3.1.1 Persistenz

Im Abschnitt 4.2.1.1 wurden die Funktionen der Komponente *Persistence* beschrieben. Messergebnisse und Feedback-Daten sollen gelesen und gespeichert werden, die später an einen Server gesendet werden. Die erste Idee war das Speichern und Lesen der Messergebnisse mit Hilfe einer Datenbank. Die Anzahl der Schritte und der Aufwand wäre hierfür jedoch insgesamt zu hoch gewesen. Außerdem müssten die Daten vor dem Senden aufbereitet werden. Deshalb wurde sich für eine leichgewichtige Lösung entschieden. Das Speichern und Lesen der Daten erfolgt im XML-Format. Hierfür wird ein Modell und eine Programmbibliothek benötigt, womit auf die Daten leicht zugegriffen werden kann. Die Programmbibliothek erstellt aus dem Modell die XML-Datei und kann umgekehrt aus der XML-Datei das Modell erstellen. Die Design-Anforderung wird in Abschnitt 4.3.1.1 erläutert. Die zusätzlich beschriebene Design-Anforderung bildet zusammen mit den Anforderungen aus Abschnitt 4.2.1.1 das Klassenmodell (s. Abb. 4.19).

**Speicherung der Daten im XML-Format** (Anhang B.1)

#### 4.3.1.2 Client-Kommunikation

Im Abschnitt 4.2.1.2 wurden die Funktionen der Komponente *ClientCommunication* beschrieben. Das mobile Endgerät verbindet sich über die IP und den Port mit dem Server über ein Socket. Es wurde sich für eine Socket-Übertragung entschieden, da sie sich leicht auf dem Android über die Java-API umsetzen lässt. Als Alternative könnte die Übertragung mit Hilfe von Webservices durchgeführt werden. Hier müssten weitere Tests erfolgen, um die Umsetzbarkeit zu überprüfen. Der Usability-Server spielt in dieser Arbeit eine untergeordnete Rolle. Auf Grund dessen wurde sich für die einfachere Variante entschieden. Vor der Übertragung an den Server werden die Dateien gepackt und anschließend versendet (s. Abschnitt 4.3.1.2). Das Packen der Dateien dient zur Minimierung des Datenvolumens bei der Übertragung. In der Systemarchitektur (s. Abschnitt 4.1) wurden die unterschiedlichen Kosten der Übertragungsarten thematisiert. Eine Minimierung des Datenvolumens verringert

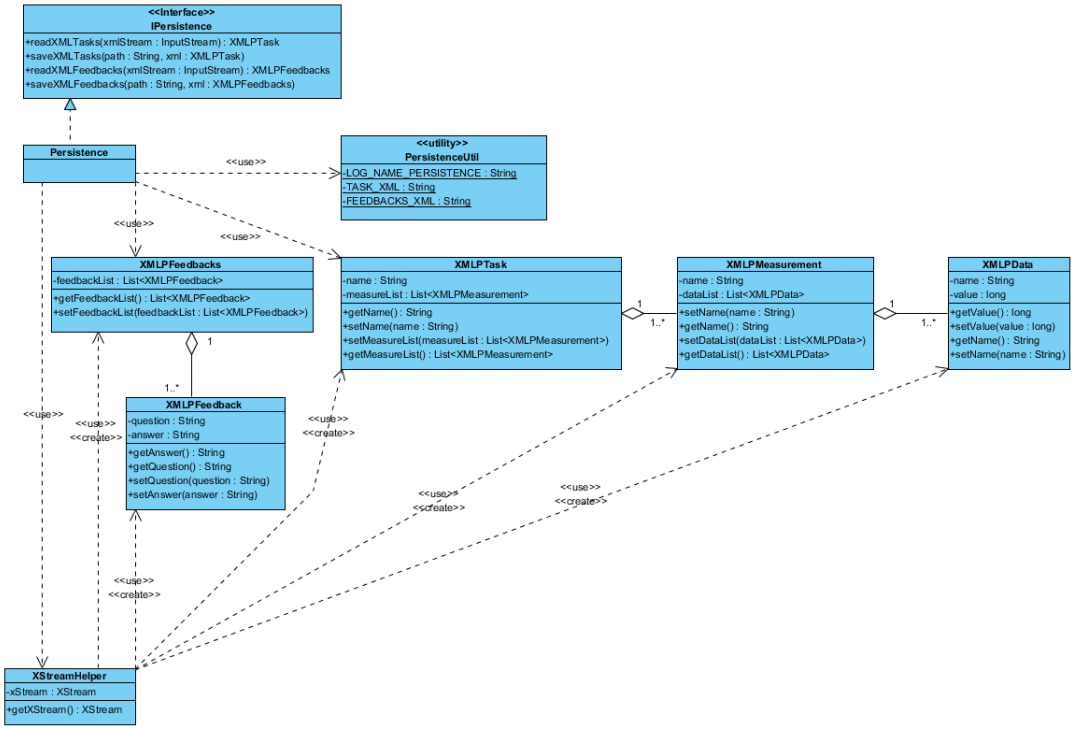



Abbildung 4.19: Klassendiagramm der Komponente *Persistence*

die Kosten und die Dauer der Übertragung. Als Packprogramm wurde das ZIP-Dateiformat gewählt, weil dies über die Android-API umgesetzt werden kann. Es wird also keine zusätzliche Programmbibliothek benötigt. Aus dieser Design-Anforderung und den Anforderungen aus Abschnitt 4.2.1.2 entsteht das Klassenmodell (s. Abb. 4.20).

**Dateien vor der Datenübertragung packen** Die Dateien werden vor der Datenübertragung im ZIP-Dateiformat gepackt.

#### 4.3.1.3 Server-Kommunikation

Im Abschnitt 4.2.1.3 wurden die Funktionen der Komponente *ServerCommunication* erläutert. Der Server wird gestartet und nimmt die Daten der mobilen Endgeräte entgegen. Für jede Datenübertragung muss der Server einen neuen Thread erstellen (s. Abschnitt 4.3.1.3); ansonsten würde aufgrund der Socket-Programmierung der Server bei einer Datenübertragung blockieren und könnte solange keine weiteren Daten entgegennehmen. Im Abschnitt 4.3.1.2 wird eine gepackte Datei im ZIP-Format übertragen. Der Server muss die gepackte Datei nach der Datenübertragung entpacken (s. Abschnitt 4.3.1.3) und die Dateien im Serververzeichnis entsprechend ablegen. Diese beiden Design-Anforderungen bilden zusammen mit den Anforderungen aus Abschnitt 4.2.1.3 das Klassenmodell (s. Abb. 4.21).

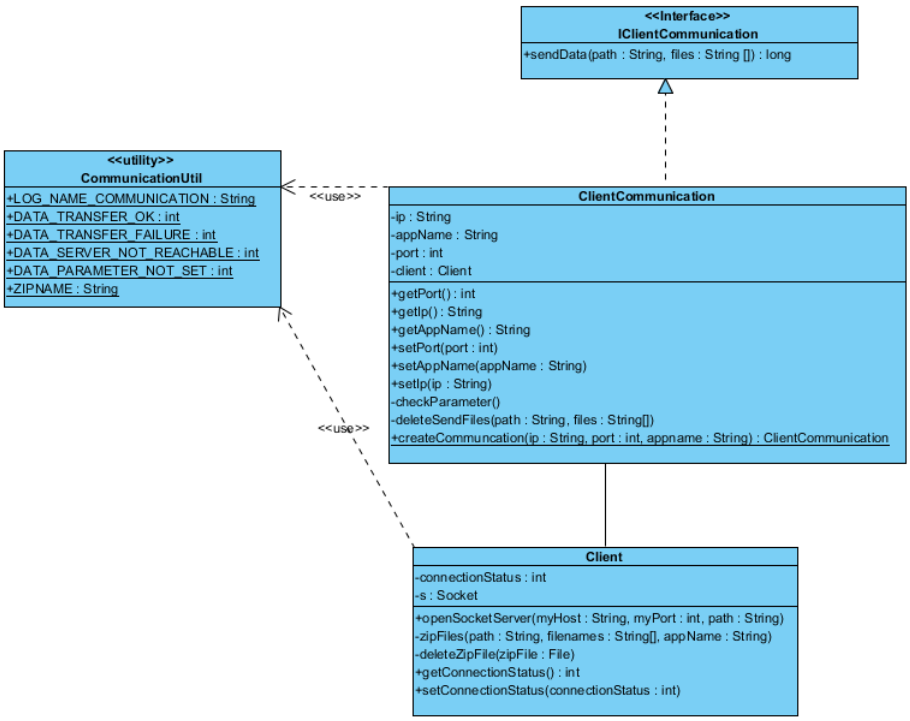



Abbildung 4.20: Klassendiagramm der Komponente *ClientCommunication*

**Für jede Datenübertragung einen Thread starten** In der Systemarchitektur (s. Abschnitt 4.1) wurde festgelegt, dass sich mehrere mobile Endgeräte mit dem Server verbinden können. Bei einer Datenübertragung vom mobilen Endgerät zum Server muss der Server auch für weitere Übertragungen verfügbar bleiben. Aus diesem Grund wird für jede Datenübertragung ein Server-Thread gestartet.

**Dateien nach Datenübertragung entpacken** Die gepackten Dateien werden nach der Datenübertragung entpackt und auf dem Server im Pfad *„Serverpfad“/“Name der Anwendung“/Dateien* abgelegt.

#### 4.3.1.4 Auswertung

Im Abschnitt 4.2.1.4 wurden die Aufgaben der Komponente *Reporting* erläutert. Der Usability-Server spielt in der Arbeit eine untergeordnete Rolle. Es wurde sich daher für eine vereinfachte Form der Auswertung entschieden. Hierfür dient eine Programmbibliothek, die Exceldateien erstellen kann. Mit dessen Hilfe wird die Auswertung der Usability-Daten für eine Anwendung erstellt. In einem weiteren Schritt oder als Alternative könnte die Auswer-

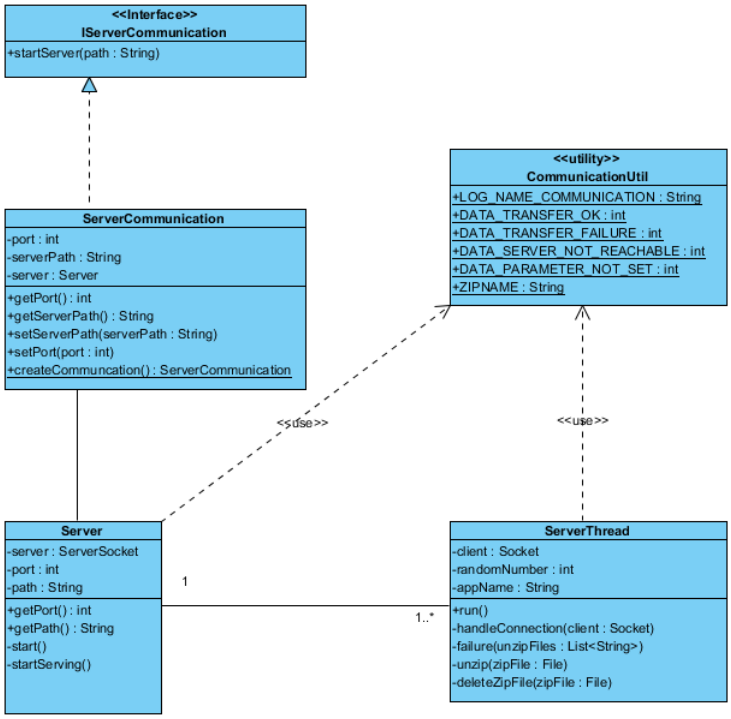



Abbildung 4.21: Klassendiagramm der Komponente *ServerCommunication*

tung mit einer Web-Applikation vorgenommen werden. Aus den Anforderungen aus Abschnitt 4.2.1.4 wurde das Klassenmodell (s. Abb. 4.22) erstellt.

#### 4.3.1.5 Messen

Im Abschnitt 4.2.1.5 wurden die Aufgaben der Komponente *Measurement* dargestellt. Folgende zusätzliche Design-Anforderungen sind notwendig:

- Mehr als eine quantitative Ausprägung für eine Messgröße (s. Abschnitt 4.3.1.5)
- Protokollierung der Touch- und Key Events über die Android-API (s. Abschnitt 4.3.1.5)
- Komplexität der Menüstruktur mit Hilfe einer Baumstruktur berechnen (s. Abschnitt 4.3.1.5)
- Completion Rate bestimmen (s. Abschnitt 4.3.1.5)

Aus den Design-Anforderungen und den Anforderungen aus Abschnitt 4.2.1.5 entsteht das Klassenmodell (s. Abb. 4.23).

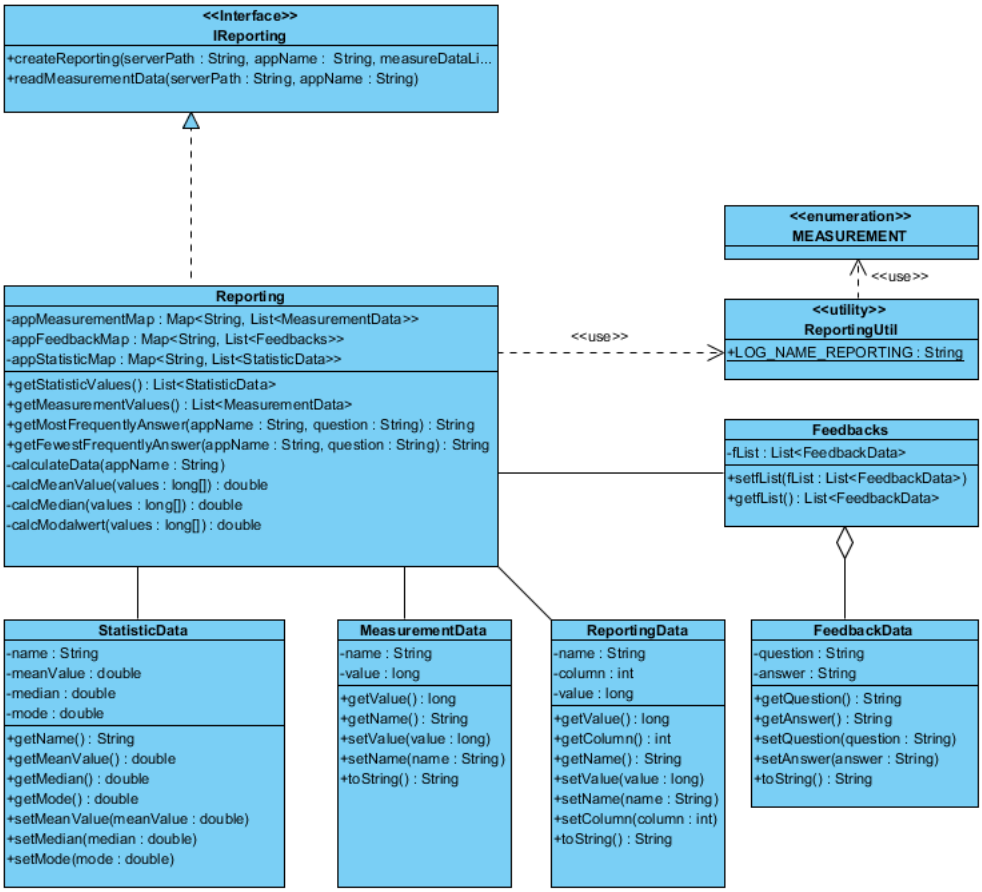



Abbildung 4.22: Klassendiagramm der Komponente *Reporting*

**Quantitative Ausprägungen einer Messgröße** Eine Messgröße kann mehrere quantitative Ausprägungen besitzen. Die Messgröße *Anzahl der Key Events* beinhaltet zum Beispiel mehrere für die unterschiedlichen Key Events. Hierfür wurde die Klasse *Data* eingeführt. Jede Messgröße hat eine Liste der Klasse *Data* als Value.

**Touch/Key Events protokollieren** Die Schnittstelle im Abschnitt 4.2.1.5 und 4.2.1.5 müssen angepasst werden, da in der Komponente *Usability* der Zugriff auf die Klassen *TouchEventManager* und *KeyEventManager* benötigt wird, die die Touch- bzw. Key Events entsprechend verwaltet.

**Komplexität berechnen** Die Activities bilden in Android die grafische Oberfläche ab. In einer Activity kann es Menü-Einträge geben, um zur nächsten Activity zu gelangen. Zu viele Verzweigungen während der Navigation in einem Programm können den Benutzer verwirren und dadurch die Benutzbarkeit der Anwendung herabsetzen. Die Anzahl der Verzweigungen in einer Anwendung sollte daher klein gehalten werden. Für die Berechnung der Komplexität wird ein Baumstruktur genutzt. Ein Knoten im Baum entspricht dabei der aktuellen Activity und die Kinder den nachfolgenden Activities. Die Tiefe und die Breite des Baums sind quantitative Ausprägungen für die Messgröße *Komplexität der Menüführung*. Die Tiefe gibt die maximale Anzahl der Activities wieder, welche beim Navigieren durch die Menüs von dem Benutzer durchlaufen werden. In einer Activity können mehrere Menü-Einträge existieren. Die Breite gibt dabei die maximale Anzahl der Menü-Einträge wieder, die während der Navigation in einer Activity aufgetreten sind.

**Completion Rate bestimmen** Für die Berechnung der *Completion Rate* muss ein perfekter Pfad für einen Task angeben werden. Welche Klassen und Methoden werden bis zum Ziel durchlaufen? Im Lebenszyklus der Anwendung werden die Methoden und Klassen hinzugefügt, die während der Task-Ausführung durchlaufen werden. Zum Schluss wird der *perfekte Pfad* mit dem *aktuellen Pfad* verglichen. Liegt die Zahl über 1 wurden mehr Schritte benötigt, bei unter 1 wurden weniger Schritte benötigt und bei genau 1 wurden die gleichen Schritte benötigt. Bei der Pfad-Überprüfung werden nur die Anzahl der Schritte verglichen. Als Alternative könnten die Methoden und Klassen in ihrer Reihenfolge verglichen werden. Mögliche Werte wären dann 1 für das Erreichen des perfekten Pfades und 0 bei irgendeiner Abweichung. Die Anzahl der Schritte hat hierbei mehr Aussagekraft, weil es nicht immer nur einen Weg geben kann, um ein Taskziel zu erreichen.

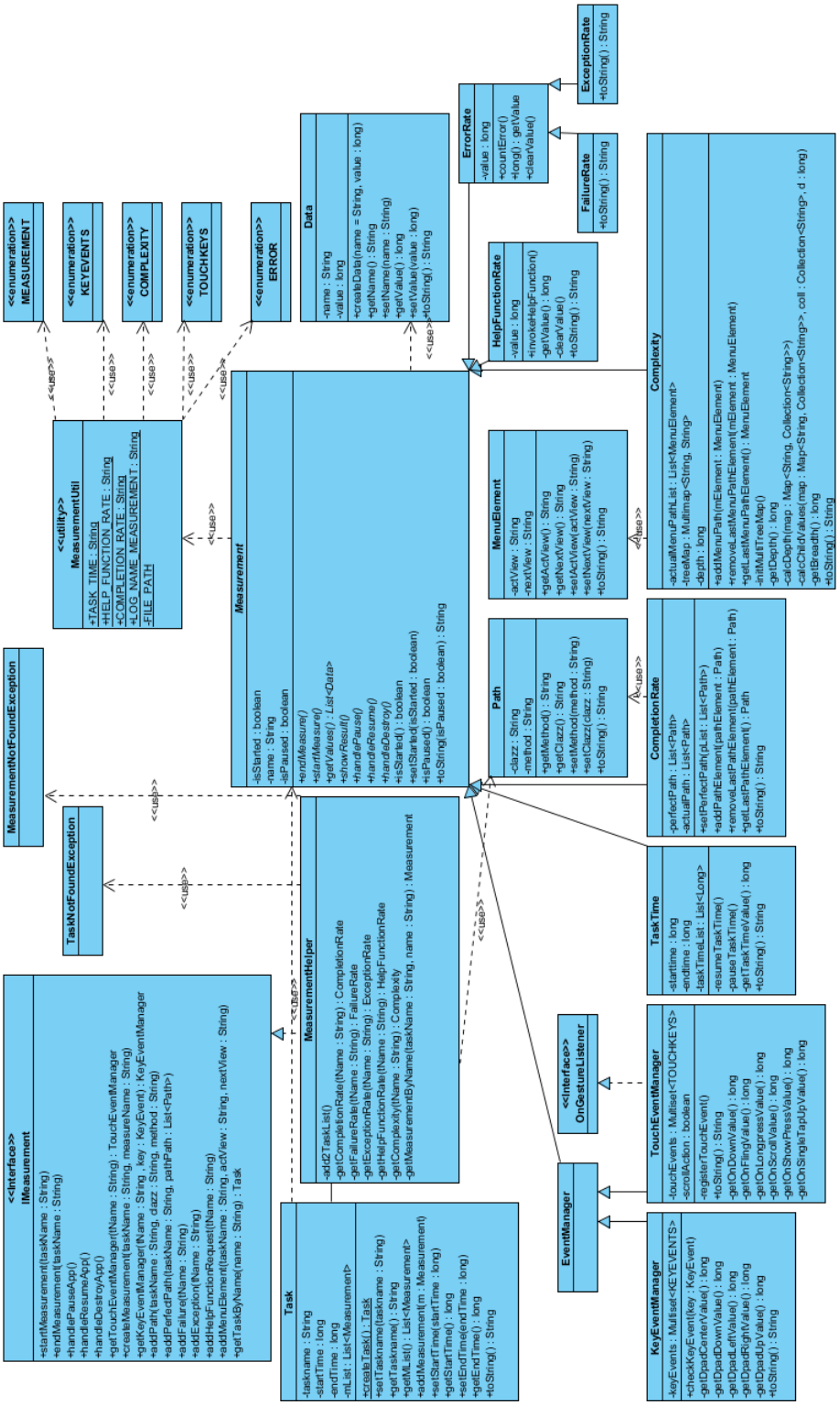



Abbildung 4.23: Klassendiagramm der Komponente *Measurement*

#### 4.3.1.6 Protokollierung

Im Abschnitt 4.2.1.6 wurden die Aufgaben der Komponente *Logging* geschildert. Aus den Anforderungen aus dem Abschnitt 4.2.1.6 wurde das folgende Klassenmodell entwickelt (s. Abb. 4.24).

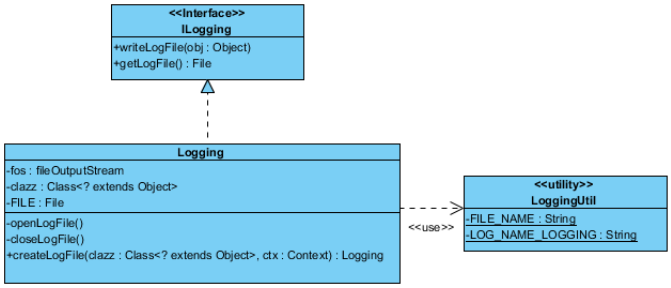



Abbildung 4.24: Klassendiagramm der Komponente *Logging*

#### 4.3.1.7 Benutzer-Interaktion

Im Abschnitt 4.2.1.7 wurden die Aufgaben der Komponente *UserInteraction* geschildert. Eine weitere Design-Anforderung wird im Abschnitt 4.3.1.7 beschrieben. Sie bildet zusammen mit den Anforderungen aus dem Abschnitt 4.2.1.7 das folgende Klassenmodell (s. Abb. 4.25).

**UserFeedback-Activity** Aus den Anforderungen geht hervor, dass ein konfigurierbarer Fragebogen angezeigt werden soll. Mit Hilfe einer Activity *UserFeedback* kann er geladen werden. Der Activity werden die entsprechenden Fragen und Antworten übergeben und entsprechend in das Design eingefügt. Hierdurch wird eine Trennung von Design und Inhalt erreicht.

#### 4.3.1.8 Konfiguration

Im Abschnitt 4.2.1.8 wurden die Funktionen der Komponente *Configuration* erläutert. Die Konfiguration wird im XML-Format vorgenommen (s. Abschnitt 4.3.1.8). XML-Dateien können einfach angepasst werden und lassen sich mit Hilfe einer Programmbibliothek für XML leicht einlesen. Diese Design-Anforderung bildet zusammen mit den Anforderungen aus Abschnitt 4.2.1.8 die Grundlage für das Klassenmodell (s. Abb. 4.26).

**Konfiguration im XML-Format** (Anhang B.2)

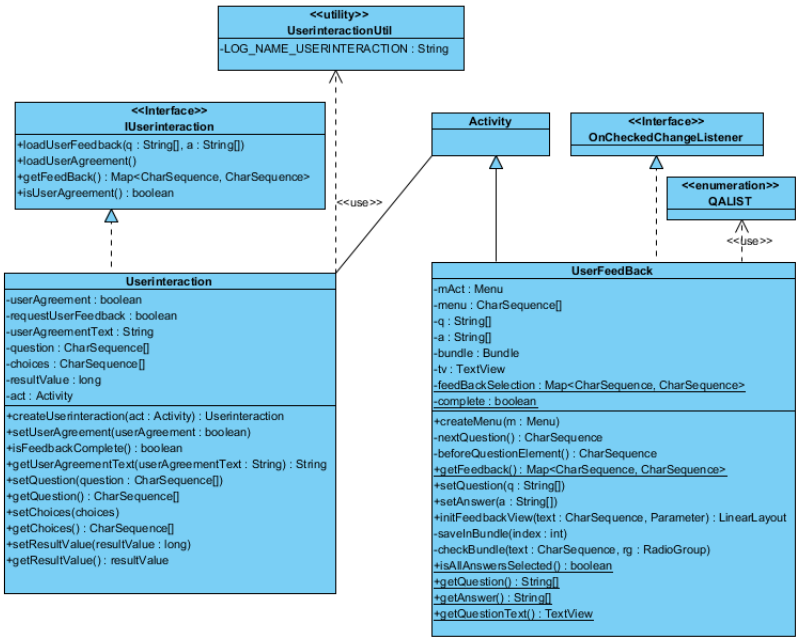



Abbildung 4.25: Klassendiagramm der Komponente *UserInteraction*

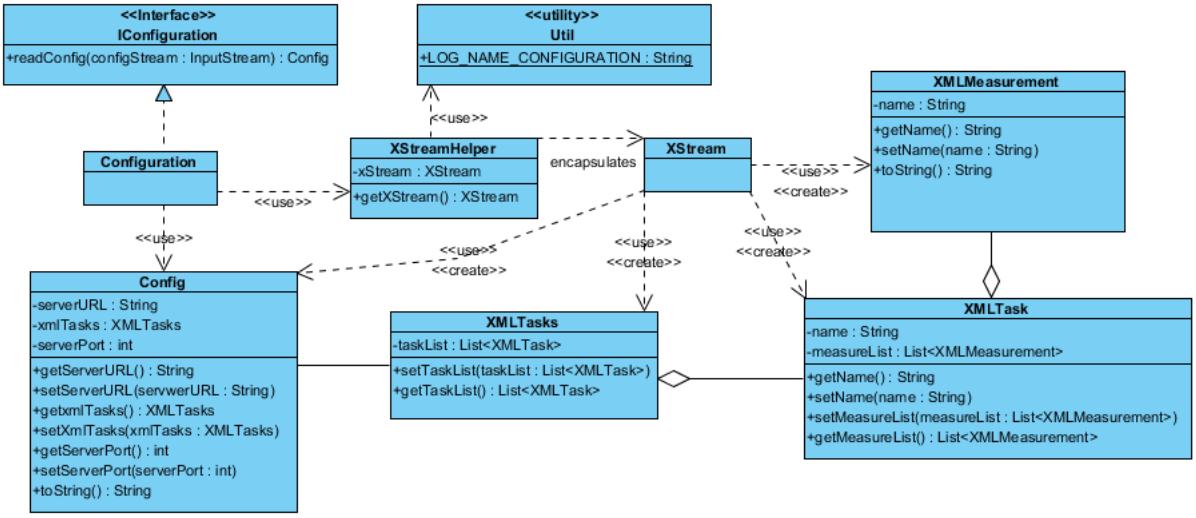



Abbildung 4.26: Klassendiagramm der Komponente *Configuration*

#### 4.3.1.9 Usability

Im Abschnitt 4.2.1.9 wurden die Funktionen der Komponente *Usability* erläutert. Das Speichern des Feedback-Fragebogens (s. Abschnitt 4.3.1.9) und der Messergebnisse (s. Abschnitt 4.3.1.9) sind weitere Design-Anforderungen. Zusätzlich müssen über die Konfiguration die entsprechenden Messgrößen eingerichtet werden (s. Abschnitt 4.3.1.9). Für das Messen von Touch- und Key Events (s. Abschnitt 4.3.1.9) wird die Klasse *UsabilityActivity* benötigt. Aus diesen Design-Anforderungen und den Anforderungen aus Abschnitt 4.2.1.9 wurde das Klassenmodell (s. Abb. 4.27) entwickelt.

**Feedback speichern** Nur beim vollständig ausgefüllten Feedback-Fragebogen werden die Feedback-Daten vor dem Senden über die Komponente *Persistence* gespeichert. Dabei werden die Daten des Fragebogens von der Komponente *UserInteraction* bezogen.

**Messgrößen speichern** Die Messgrößen werden in der Komponente *Measurement* verwaltet. Die Daten werden nach dem Beenden der Messung über die Komponente *Persistence* gespeichert. Hiermit wird die nicht-funktionale Anforderung (s. Kapitel 3.3.2) erfüllt.

**Messgrößen mit der Konfiguration einrichten** Die Konfiguration wird mit der Komponente *Configuration* eingelesen. Mit Hilfe der Daten werden die Messgrößen und Tasks in der Komponente *Measurement* erstellt.

**Messen von Touch- und Key Events über die *UsabilityActivity*** Über die API einer Activity können Touch- und Key Events abgefangen und verarbeitet werden. Aus diesem Grund wurde die *UsabilityActivity* entworfen. Alle grafischen Oberflächen in einer Android-Anwendung werden von der Klasse Activity abgeleitet. Sie müssen alle durch die *UsabilityActivity* ersetzt werden. Die *UsabilityActivity* wird der *Touch-* und *KeyEventManager* übergeben. Die Touch- und Key Events werden in dieser abgefangen und in den entsprechenden Managern protokolliert.

### 4.3.2 Mobiles Endgerät

In Abschnitt 4.2.2 wurde das Komponentenmodell und die wichtigen Schnittstellen beschrieben. Die Komponente *Client* kapselt dabei die Schnittstellen der Komponente *Usability*. Daraus ergibt sich das folgende Klassenmodell (s. Abb. 4.28). In dem Abschnitt 4.3.2.1 wird die Interaktion zwischen den verschiedenen Komponenten gezeigt.

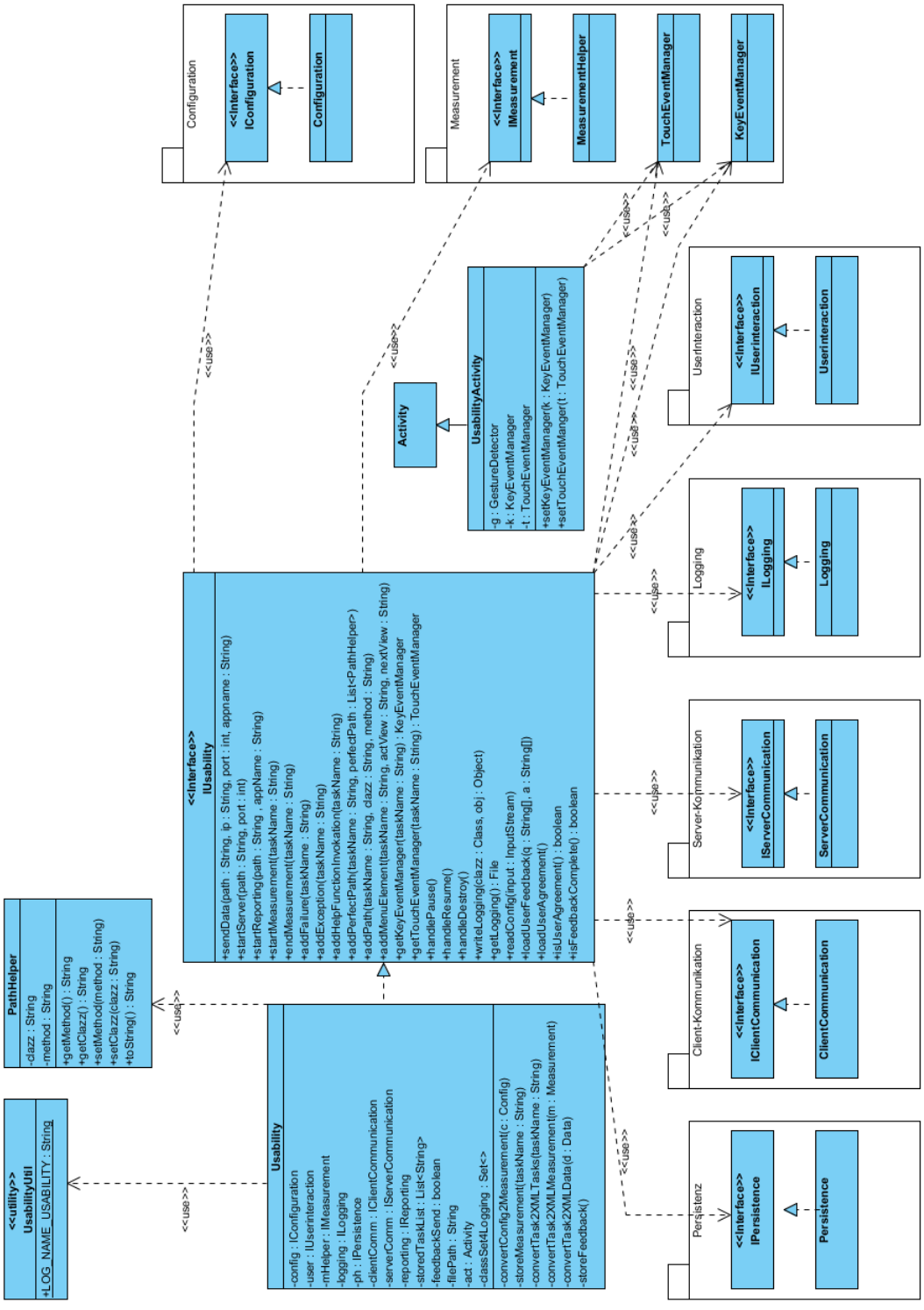



Abbildung 4.27: Klassendiagramm der Komponente *Usability*

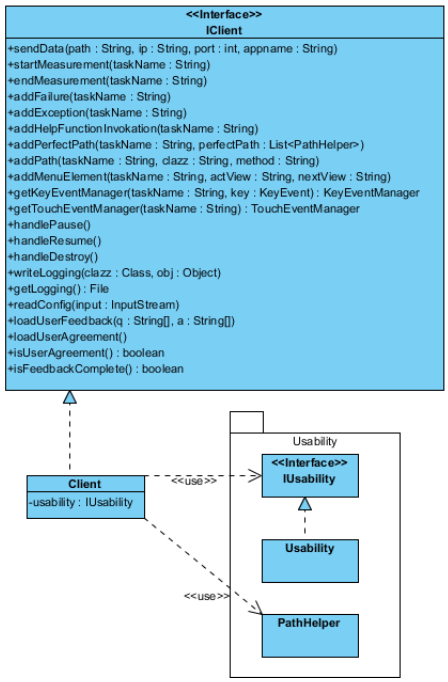



Abbildung 4.28: Klassendiagramm der Komponente *Client*

#### 4.3.2.1 Interaktion der Komponenten

Die Abb. 4.29 zeigt die Interaktion zwischen der Komponente *Client* und den zugehörigen Komponenten. Im ersten Schritt muss die mobile Anwendung die Konfiguration einlesen. Das Einlesen erfolgt über die Komponenten *Usability* (1.1)[2] und *Konfiguration* (1.1.1). Die Komponente *Usability* erstellt mit Hilfe der eingelesenen Konfiguration die Messgrössen (1.1.3) in der Komponente *Messen*. Im nächsten Schritt wird die Benutzer-Zustimmung für die Usability-Untersuchung erbeten. Dabei werden die Komponenten *Client* (2), *Usability* (2.1) und *Benutzer-Interaktion* (2.1.1) hintereinander aufgerufen. Das Ergebnis der Zustimmung wird über *Client* (3), *Usability* (3.1) und *Benutzer-Interaktion* 3.1.1 abgefragt. Falls die Zustimmung positiv war, wird die Messung über die Komponenten *Client* (5), *Usability* (5.1) und *Messen* (5.1.1) gestartet. Nach dem Start der Messung können diese bearbeitet werden. Hierzu gehören das Hinzufügen von Fehlern sowie Menü- oder Methodenaufrufe. Außerdem können die Messungen auf Zustandsänderungen (Pause, Destroy, Resume) entsprechend reagieren. Diese Aufrufe erfolgen über die Komponenten *Client*, *Usability* und *Messen*. Im Verlauf der Messung kann außerdem eine Protokollierung angelegt werden. Der Aufruf erfolgt in der Reihenfolge *Client* (6), *Usability* (6.1) und *Protokollierung* (6.1.1). Zum Ende hin wird die Messung über *Client* (7), *Usability* (7.1) und *Messen* (7.1.1) gestoppt. Danach wird das Ergebnis der Messungen über die Komponente *Usability* (7.1.2) für die Speiche-

[2] Die Nummerierung kann im Sequenzdiagramm Abb. 4.29 abgelesen werden

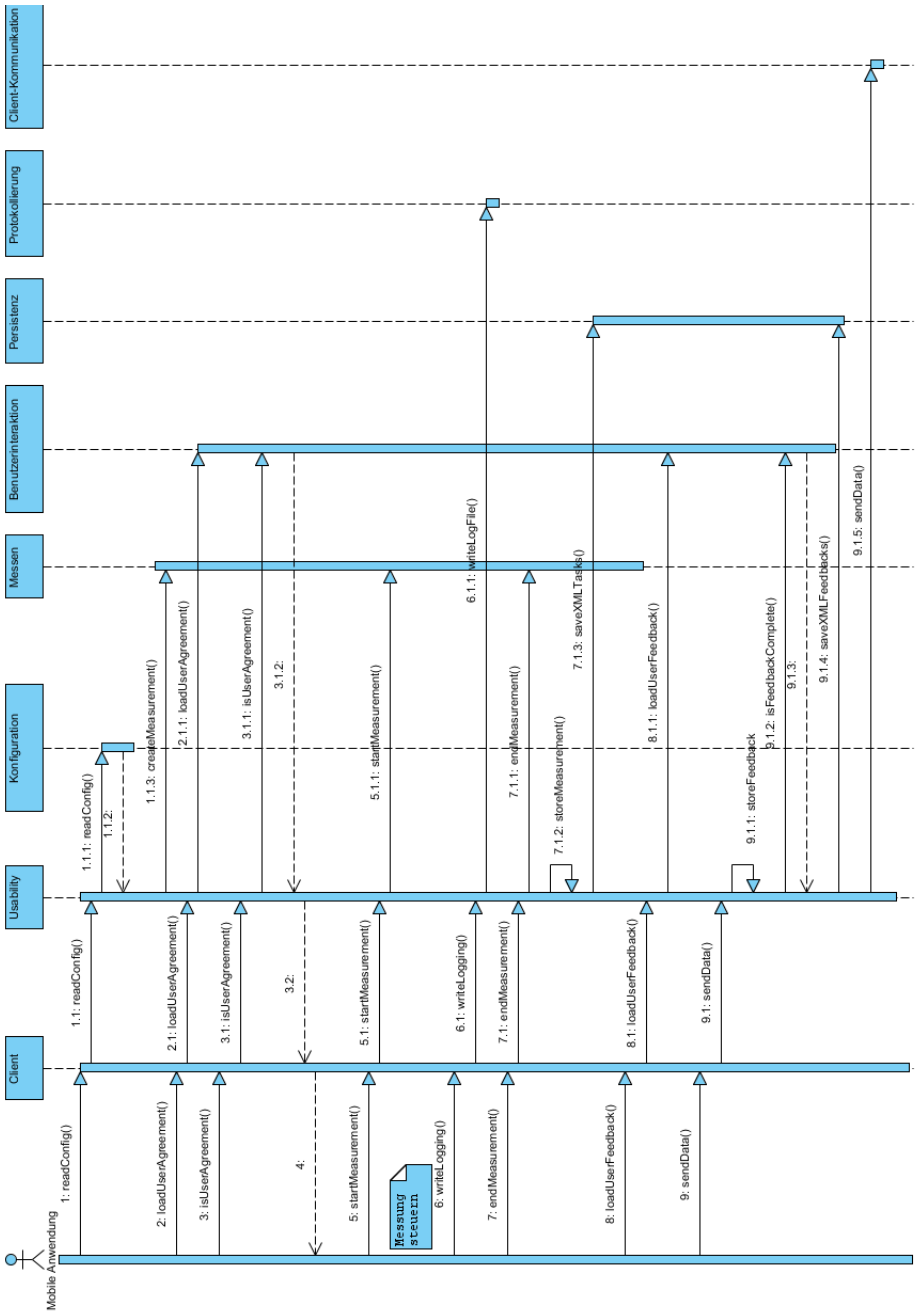



Abbildung 4.29: Sequenzdiagramm der Komponente *Client* mit den zugehörigen Komponenten

rung vorbereitet. Anschließend wird über *Persistenz* (7.1.3) die Messung gespeichert. Zum Abschluss der Usability-Untersuchung kann ein konfigurierbarer Feedback-Fragebogen geladen werden. Dieser Aufruf erfolgt über die Komponenten *Client* (8), *Usability* (8.1) und *Benutzer-Interaktion* (8.1.1). Zu guter Letzt werden die Daten an den Usability-Server gesendet. Über die Komponente *Client* (9) wird die Komponente *Usability* (9.1) angesprochen. Dort werden die Feedback-Daten, falls vorhanden, für die Speicherung vorbereitet (9.1.1). Es wird über die Komponente *Benutzer-Interaktion* (9.1.2) abgeprüft, ob der Feedback-Fragebogen komplett ausgefüllt wurde. Führt die Abfrage zu einem positiven Ergebnis, werden die Daten mit Hilfe der Komponente *Persistence* (9.1.4)[3] gespeichert. Nachfolgend werden die Daten über die *Client-Kommunikation* (9.1.5) gesendet.

### 4.3.3 Usability-Server

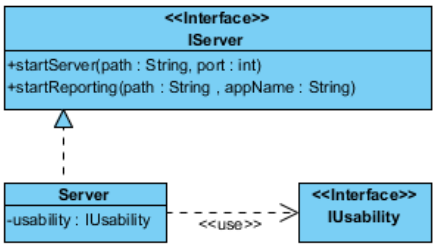



Abbildung 4.30: Klassendiagramm der Komponente *Server*

In Abschnitt 4.2.3 wurde das Komponentenmodell und die wichtigen Schnittstellen beschrieben. Die Komponente *Server* kapselt dabei die Schnittstellen der Komponente *Usability*. Daraus resultiert das folgende Klassenmodell (s. Abb. 4.28). Im Abschnitt 4.3.3.1 wird die Interaktion zwischen den verschiedenen Komponenten gezeigt.

#### 4.3.3.1 Interaktion der Komponenten

Das Diagramm 4.31 zeigt den Ablauf der Komponente *Server* mit den zugehörigen Komponenten. Im ersten Schritt wird der Usability-Server über die Komponenten *Server* (1), *Usability* (1.1) und *Server-Kommunikation* (1.1.1)[4] gestartet. Anschließend kann der Server die Daten der mobilen Endgeräte entgegennehmen. Im weiteren Verlauf kann die Auswertung der Mess- und Feedback-Daten gestartet werden. Die Komponente *Server* (2) leitet die Anfrage

[3] Die Nummerierung kann im Sequenzdiagramm Abb. 4.29 abgelesen werden
[4] Die Nummerierung kann im Sequenzdiagramm Abb. 4.31 abgelesen werden

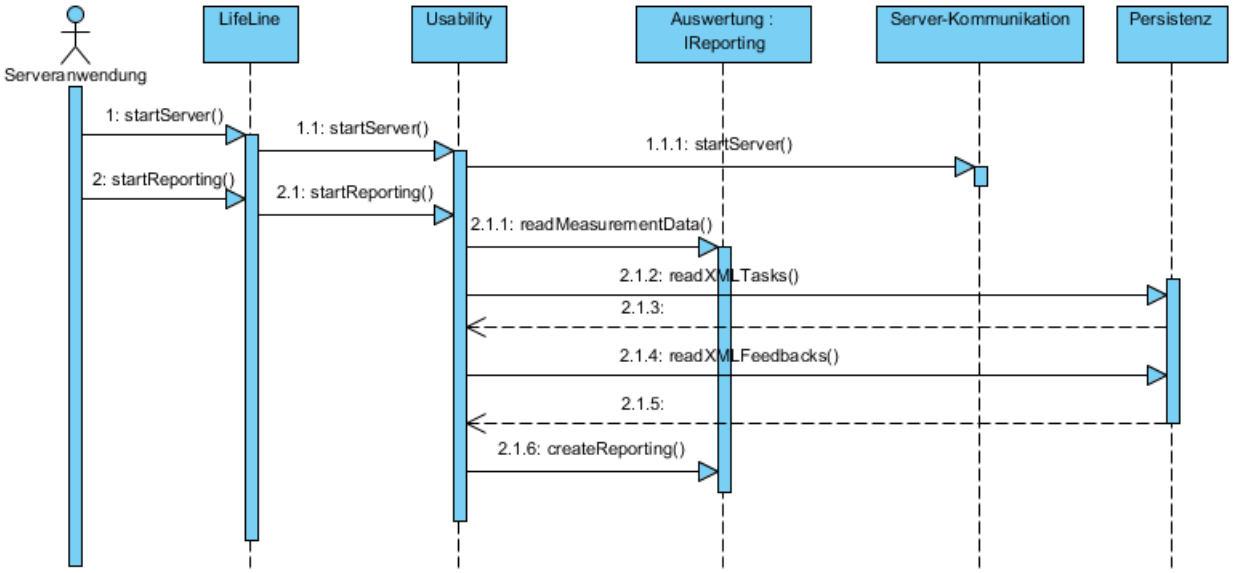



Abbildung 4.31: Sequenzdiagramm der Komponente *Server* mit den zugehörigen Komponenten

an die Komponente *Usability* (2.1) weiter. Hier werden die XML-Dateien mit den Messergebnissen und Feedback-Daten aus dem Server-Verzeichnis der entsprechenden Anwendung eingelesen (2.1.1). Anschließend werden die Messergebnisse über die Komponente *Persistence* (2.1.2) ausgelesen. Ebenfalls werden die Feedback-Daten über die Komponente *Persistence* (2.1.4) ermittelt. Zum Schluss wird eine Auswertungsdatei (Excel) für die Anwendung über die Komponente *Auswertung* (2.1.6)[5] erstellt.

## 4.4 Zusammenfassung

In diesem Kapitel wurden aus den Anforderungen aus Kapitel 3.2.1 ein Entwurf einer Usability-Komponente präsentiert. Darüber hinaus dessen Einsatz auf dem mobilen Endgerät und dem Server. Dazu werden folgende Themengebiete betrachtet:

- Systemarchitektur
  Hier wurden die Hardware- und Softwarekomponenten vorgestellt, die auf dem mobilen Endgerät und dem Server zum Einsatz kommen, insbesondere die Verwendung der relevanten Usability-Komponente auf beiden Systemen.

[5] Die Nummerierung kann im Sequenzdiagramm Abb. 4.31 abgelesen werden

- Anwendungsarchitektur
  Die *Usability-Komponente* wurde mit Hilfe des fachlichen Subsystemmodells in geschlossene Komponenten unterteilt, die über ihre Schnittstellen kommunizieren. Eine wichtiger Aspekt spielte hierbei die Anpassung und Kapselung der Schnittstellen für das mobile Endgerät und den Server.

- Feinentwurf
  Im Feinentwurf wurden die Komponenten und ihre Schnittstellen weiter verfeinert und ergänzt. Zusätzlich wurde das Zusammenspiel der Komponenten für das mobile Endgerät und den Usability-Server im Detail erläutert.

Im nächsten Kapitel wird auf Basis der Systemarchitektur, der Anwendungsarchitektur und dem Feinentwurf ein konkreter Prototyp entwickelt.

# 5 Realisierung

In diesem Kapitel wird eine konkrete Realisierung der Usability-Komponente aus dem Kapitel 4 durch die Implementierung eines Prototyps vorgestellt. Er wird auf Basis der definierten Systemarchitektur (s. Kapitel 4.1), der Softwarearchitektur (s. Kapitel 4.1) und dem Feinentwurf (s. Kapitel 4.3) entworfen. Zusätzlich wird ein Anwendungsbeispiel genommen, welche die Usability-Komponente implementiert. Anhand eines Beispiel-Szenarios wird dies dann erprobt.

## 5.1 Implementierung des Prototyps

Der Abschnitt behandelt den Umfang der Implementierung, beschreibt die verwendenten Programmbibliotheken und präsentiert die Realisierungsdetails.

### 5.1.1 Implementierungsumfang

Ziel der Arbeit ist eine prototypische Realisierung des Entwurfs einer Usability-Komponente im Kontext von Android. Hierbei wurde der Entwurf aus Kapitel 4 nahezu komplett umgesetzt. Als Ergebnis kommt die Usability-Komponente mit Hilfe eines Prototyps auf dem mobilen Endgerät und dem Usability-Server zum Einsatz; die Komponente erfüllt sämtliche funktionale Anforderungen (s. Kapitel 3.2.1). Auf dem Usability-Server wurde die Präsentationsschicht nicht umgesetzt. In diesem Fall wurde eine vereinfachte Form zur Darstellung der Ergebnisse verwendet. In den Realisierungsdetails im Abschnitt 5.1.3 wird dies genauer beschrieben. Die Komponenten *Persistence*, *ClientCommunication*, *ServerCommunication*, *Reporting*, *Measurement*, *Protokollierung*, *UserInteraction*, *Configuration*, *Usability* wurde wie im Entwurf spezifiziert realisiert. Die Komponenten *Client* und *Server* werden hierbei nicht genauer betrachtet, da sie nur für die Kapselung der Schnittstellen der Komponente *Usability* zuständig sind.

### 5.1.2 Verwendete Software-Komponenten

In den folgenden Abschnitten werden die eingesetzten Programmbibliotheken für die Implementierung beschrieben.

#### 5.1.2.1 Android

Die Android Programmbibliothek in der Version 1.6 wurde in der Realisierung verwendet [Google (2010)]. Zu Beginn dieser Arbeit war die Version 1.6 die aktuelle Android-API. Außerdem war auf den zur Verfügung stehenden mobilen Endgeräten auch jeweils die Android-API 1.6 verfügbar. Die zukünftigen Versionen von Android sollen abwärtskompatibel sein. In der Entwicklung muss man festlegen, welche kleinste Version für die Anwendung notwendig ist. Ebenso sollten keine Methoden einer höheren Version verwendet werden.

#### 5.1.2.2 Google Collections Library 1.0

Ein elementarer Bestandteil in Java Projekten ist das Java Collections Framework. Google erweitert mit Hilfe der Google Collections Library dieses Framework. Diese wurde Ende Dezember 2009 in der Version 1.0 veröffentlicht [Google (2009)].

#### 5.1.2.3 XStream

XStream ist eine einfache Bibliothek zum Serialisieren von Objekten nach XML und wieder zurück. [XStream (2008)].

#### 5.1.2.4 Apache POI

Apache POI ist eine Java-API zum Erstellen und Bearbeiten von Dateiformaten, die auf dem Office Open XML Standard (OOXML) und dem Microsoft-Dateiformat *OLE-2 Compound Document* beruhen. Dateien in diesem Format sind unter anderem die meisten Microsoft-Office-Dateien wie zum Beispiel Excel- und Word-Dateien [POI (2009)].

### 5.1.3 Implementierungsdetails

In diesem Abschnitt werden die Realisierungsdetails ausgesuchter Komponenten des Entwurfes beschrieben.

#### 5.1.3.1 Messung

Die Komponente *Measurement* spielt eine zentrale Rolle für die Messung von Usability. Sie setzt dabei alle Anwendungsfälle unterhalb der Anwendungsfall-Gruppe *Messen* in Kapitel 3.2.1.2 um. Für die Realisierung der Anforderung *Completion Rate* und *Komplexität der Menüführung* wurde eine vereinfachte Lösung gewählt. Für beide Messgrößen sind Daten aus der Anwendung notwendig. Für die Berechnung der Messgröße *Completion Rate* müssen die Methodenaufrufe und die zugehörigen Klassen in ihrem Ablauf der Messgröße mitgeteilt werden. In der Umsetzung muss der Entwickler dies manuell hinzufügen. Ebenso müssen für die Berechnung der Messgröße *Komplexität der Menüführung* die Menüwechsel hinzugefügt werden. Es wäre wünschenswert, diese Schritte zu automatisieren. Die Berechnung der Komplexität wird mit Hilfe einer Baumstruktur bestimmt. An dem Baum kann dann die Tiefe und Breite jeweils als quantitative Ausprägung abgelesen werden. Der Baum wurde dabei mit Hilfe der TreeMultiMap von der *Google Collections Library 1.0* (s. Abschnitt 5.1.2.2) umgesetzt. In einem Menü haben alle Einträge den gleichen aktuellen View aber verschiedene nachfolgende Views. Die TreeMultiMap-Implementation erlaubt für einen Schlüssel (aktueller View) verschiedene Werte (nachfolgende Views).

#### 5.1.3.2 Auswertung

Für den Usability-Server wurde nur die Auswertung umgesetzt. Da es sich um keine vollständige Server-Anwendung mit einer Präsentationsschicht handelt, wurde eine vereinfachte Form der Auswertung gewählt. Mit der API von *Apache POI* (s. Abschnitt 5.1.2.4) kann eine Auswertungsdatei in Form von Excel für jede Anwendung erstellt werden. Dabei werden in der Komponente *Reporting* die Messergebnisse, Feedback-Daten und Statistiken in die Datei aufgenommen. Die Abb. 5.1 zeigt den Aufbau dieser Datei.

#### 5.1.3.3 Konfiguration

Mit Hilfe der XStream-Api (s. Abschnitt 5.1.2.3) werden die Elemente der Konfigurationsdatei in einem Modell abgebildet:

- XML-Element: *config* => Klasse: *Config*
- XML-Element: *tasks* => Klasse: *XMLTasks*
- XML-Element: *task* => Klasse: *XMLTask*
- XML-Element: *measurement* => Klasse: *XMLMeasurement*

Mit dem Modell werden die XML-Elemente auf deren Klassen abgebildet.


Abbildung 5.1: Auszug aus der Exceldatei für die Auswertung

#### 5.1.3.4 Persistenz

Für das Lesen und Speichern der XML-Dateien der Messergebnisse und Feedback-Daten wird die XStream-API (s. Abschnitt 5.1.2.3) verwendet. Die Elemente der XML-Dateien für die Messergebnisse und der Feedback-Daten werden in einem Modell abgebildet:

**Messergebnisse**

- XML-Element: *task* => Klasse: *XMLPTask*
- XML-Element: *measurement* => Klasse: *XMLPMeasurement*
- XML-Element: *data* => Klasse: *XMLT*

**Feedback-Daten**

- XML-Element: *feedbacks* => Klasse: *XMLPFeedbacks*
- XML-Element: *feedback* => Klasse: *XMLPFeedback*

Mit dem Modell werden die XML-Elemente auf deren Klassen abgebildet.

#### 5.1.3.5 Benutzer-Interaktion

In der Komponente *UserInteraction* wird der Feedback-Fragebogen in einer Activity aus der Android-API (s. Abschnitt 5.1.2.1) abgebildet. Der Inhalt des Fragebogens wird der Activity übergeben und diese erstellt den Fragebogen anhand der angegebenen GUI. Die grafischen Oberflächen werden normalerweise in einer XML-Datei dargestellt. Die Usability-Komponente wird aber später als Programmbibliothek eingebunden und dort lassen sich zur Laufzeit aus den XML-Dateien keine Views erstellen. Deshalb wurden die grafischen Elemente in Java beschrieben (s. Anhang B.3). Das Layout besteht aus einem *LinearLayout*, das ein weiteres *LinearLayout* enthält. Das innere *LinearLayout* enthält ein *TextView* für die Frage und eine RadioGroup mit den zugehörigen *RadioButtons* für die Antworten. Die Menü-Einträge der Activity sind *Next*, *End* und *Back*. Die Abb. 5.2 zeigt die Darstellung des Fragebogens auf dem mobilen Endgerät.

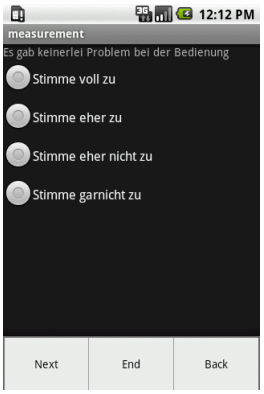


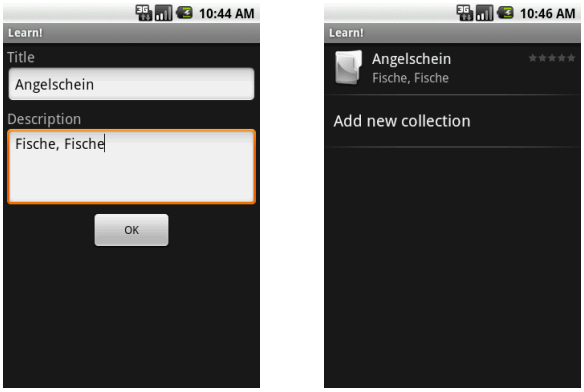

Abbildung 5.2: Fragebogen

## 5.2 Anwendungsbeispiel

In diesem Abschnitt wird die Anwendung *Learn!* [Vaitukaitis (2010)] vorgestellt. Learn! ist eine Android-Applikation, die im Rahmen des *„Summary placement Project“* an der Cambrige Universität entwickelt wurde. Mit Learn! können so genannte Collections (Themen) mit Hilfe von Flashcards (Lernkarteien) erlernt werden.

### 5.2.1 Szenario 1: Erstellen von Flashcards

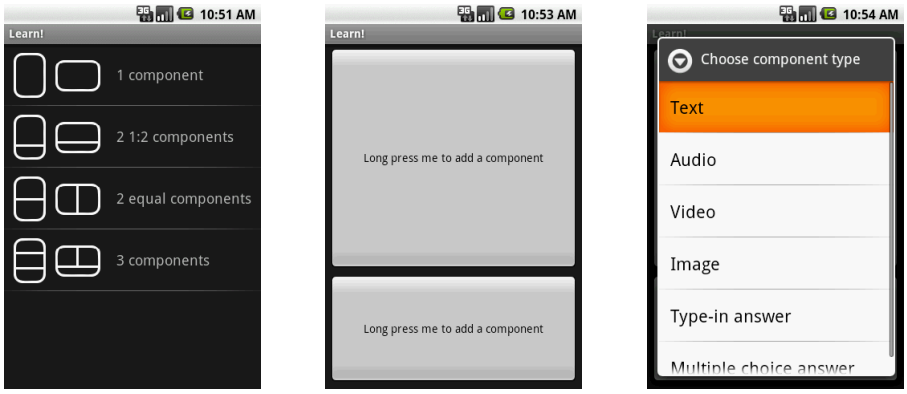
In Learn! kann man für eine Collection (Thema) eigene Flashcards (Lernkarteien) erstellen. Hierfür wählt man im Hauptmenü den *Editor* aus, danach erscheint eine Liste der erstellen




(a) (b)

Abbildung 5.3: Erstellen einer Collection in *Learn!*

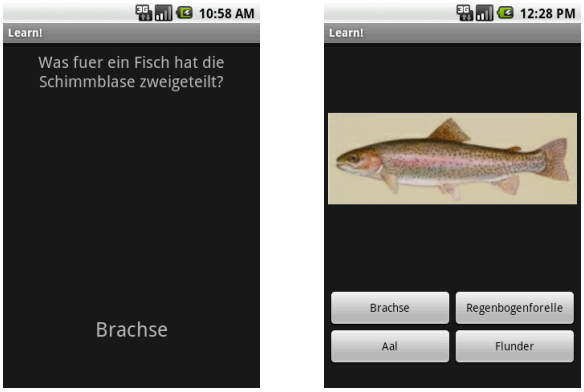
Collections. Dort kann man ebenfalls eigene Collections anlegen. Im ersten Schritt wird eine Collection mit dem Titel *Angelschein* und einer Beschreibung *Fische, Fische* erstellt. Mit *OK* bestätigen wir das Hinzufügen. Anschließend befindet sich die neue Collection in der Liste (s. Abb. 5.3).




(a) (b) (c)

Abbildung 5.4: Erstellen einer Flashcard *Learn!*

Bei Auswahl einer Collection kann im nachfolgenden View die Liste der Flashcards für diese Collection angeschaut werden. Die Collection *Angelschein* wurde neu angelegt und hat deshalb auch keine Flashcards. Mit *Add new Card* können neue Flashcards hinzugefügt werden. Im darauf folgenden Schritt wählt man die Art der Flashcard aus. In diesem Szenario wird der zweite Punkt *1:2 Components* in der Liste ausgewählt. Mit einem *Longpress* auf die jeweilige Komponente erscheint eine Auswahl, welche Art von Komponente hinzugefügt werden soll. Für die obere und untere Komponente wird *Text* gewählt (s. Abb. 5.4).




(a) (b)

Abbildung 5.5: Flashcard Beispiele in *Learn!*

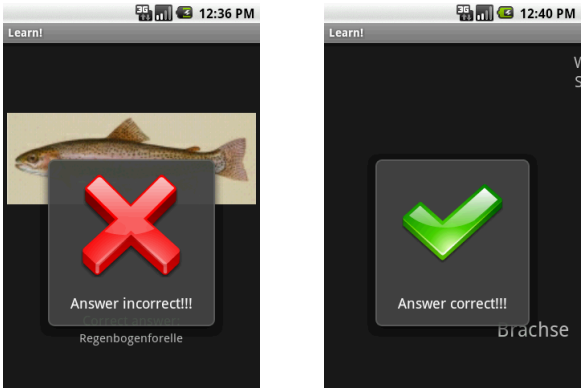
Die Abb. 5.5 zeigt beispielhaft die beiden erstellten Flashcards mit Fragen aus der Collection *Angelschein*. Für die zweite wurde auch *1:2 Components* ausgewählt. Die obere Komponente ist hierbei ein Bild (Image). Es ist nur möglich das Bild direkt über die Kamera als Snapshot einzubinden oder aus dem Verzeichnis für die Kamera ein Bild auszuwählen. Für die untere Komponente wird *Multiple Choice Answer* gewählt. Anschließend erscheinen in einer View vier Eingabefelder für die Antworten mit einem Radio-Button für die richtige Antwort.

### 5.2.2 Szenario 2: Lerneinheit durchführen

Im zweiten Szenario wird im Hauptmenü *Learn* ausgewählt. Anschließend wird die Lerneinheit für die erstellte Collection *Angelschein* gestartet. Die Flashcard nur mit Text wird so angezeigt wie in Abb. 5.5a dargestellt und die Flashcard mit dem Bild und den Antwort-Möglichkeiten als *Multiple Choice Answer* wie in Abb. 5.5b. Mit einer Auswahl der Buttons wird überprüft, ob die Antwort falsch oder richtig ist. War die entsprechende Auswahl falsch wird ein rotes Kreuz mit dem Text *Answer incorrect* angezeigt, und anschließend die korrekte Antwort gegeben. Bei Auswahl der korrekten Antwort wird ein grüner Haken mit dem Text *Answer correct* dargestellt und zur nächsten Flashcard gewechselt (s. Abb. 5.6).

## 5.3 Integration des Prototyps in das Anwendungsbeispiel

Im Kapitel 5.2 wurde die Anwendung *Learn!* vorgestellt. Anschließend wurden zwei typische Szenarien beschrieben. Der Entwurf wird anhand dieser Szenarien durch die Integration des Prototyps realisiert. Zum besseren Verständnis wird mit Hilfe von Codefragmenten die




(a) Falsche Antwort (b) Korrekte Antwort

Abbildung 5.6: Lernen mit den Flashcards in *Learn!*

Realisierung beschrieben. Der vollständige Code zur Anwendung wird hier nicht im Detail aufgeführt, weil dieser zu diesem Zeitpunkt für eine Freigabe nicht bestimmt war.

### 5.3.1 Konfiguration

Im ersten Schritt wird die Konfiguration der Usability-Komponente vorgenommen. Die beiden Szenarien bilden dabei die Grundlage zur Bestimmung der Tasks in dem XML-File. Innerhalb der Tasks werden die relevanten Messgrössen bestimmt (s. Anhang B.4). Als mögliche Messgrößen wurden *Task Time*, *Anzahl der Key Events* und *Anzahl der Touch Events* für das erste Szenario festgelegt. Für das zweite wurde zusätzlich noch die *Komplexität der Menüführung* hinzugenommen. Der Task *allgemein* beinhaltet die Messgrößen *Aufruf der Hilfefunktion* und *Fehleranzahl*. Dieser Task deckt die Messgrößen ab, welche für die gesamte Anwendung gelten sollen. Anschließend muss die Konfiguration in das Ressourcen-Verzeichnis der Anwendung *Learn!* abgelegt werden. Nach dem Einbinden der entsprechenden Libraries für das mobile Endgerät muss die AndroidManifest.xml angepasst werden. Die *UserFeedback-Activity* muss in dieser Manifest-Datei hinzugefügt werden, um später den Feedback-Fragebogen zu starten.

```
< activity  android:name="de.vollmer.usability.userinteraction.UserFeedBack"></activity>
```

Das Senden der Daten sollte in unregelmäßigen Abständen erfolgen, um eine entsprechende Datenmenge angesammelt zu haben. Der Benutzer sollte dann das Senden der Daten bestätigen oder einen entsprechenden Hinweis erhalten, um dies manuell zu erledigen. In

*Learn!* kann man unter dem Menüpunkt *Share Collections* up- und downloaden. Hier besteht die Möglichkeit, einen weiteren Menüpunkt einzurichten, um die Usability-Ergebnisse später manuell hochzuladen. Ein entsprechender Service muss dafür in der Manifest-Datei beschrieben und entwickelt werden.

```
<service android:name="UploadUserStatistic"></service>
```

Die Abb. 5.7 zeigt den weiteren Menüpunkt unter Share und den Ablauf des Upload-Services als *Status Bar Notifications*.

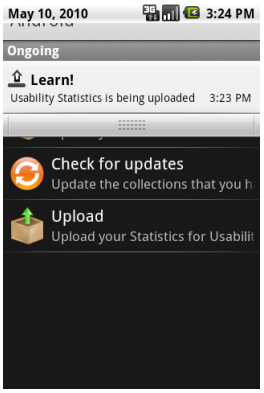



Abbildung 5.7: Upload der Usability-Ergebnisse

### 5.3.2 Integration

In der Main-Activity wird die Komponente *Client* mit der Konfiguration initialisiert (s. Abb. 5.1)

```
//Abfragen des Wertes aus den Shared Preferences
 SharedPreferences prefs = PreferenceManager.
    getDefaultSharedPreferences(this);
 agreement = prefs.getBoolean("agreement", false);
 FileInputStream config = null;
 try {
   // Einlesen der Konfiguration
   config = openFileInput("config.xml");
 } catch (FileNotFoundException e) {
   e.printStackTrace();
 }
 //Client Initialisieren
```

```
    IClient client = new Client(this, config);
}
```

Listing 5.1: Client mit Konfiguration initialisieren

Anschließend wird die Einverständnis-Erklärung für den Benutzer geladen, wenn diese noch nicht abgefragt worden ist (s. Abb. 5.2).

```
    if (!agreement) {
      client.loadUserAgreement();
    }
```

Listing 5.2: Einverstädniserklärung starten

Das Ergebnis wird in einer so genannten *Shared Preferences* gespeichert. Hiermit wird sichergestellt, ob die Abfrage schon gestellt wurde und mit welchem Ergebnis.
Der allgemein Task wird im Hauptmenü gestartet. Der Task beinhaltet Messgrößen, die für die gesamte Anwendung gelten (s. Abb. 5.3).

```
client.startMeasurement("allgemein");
```

Listing 5.3: Starten der Messung für den Task „allgemein“ initialisieren

Beim Aufruf der Menüpunkte *Learn* oder *Editor* wird die Messung gestartet (s. Abb. 5.4).

```
//Menüpunkt Learn
client.startMeasurement("task2");
//Menüpunkt Editor
client.startMeasurement("task2");
```

Listing 5.4: Starten der Messung

Während die Messung läuft, können einige Messgrößen wie *Aufruf der Hilfefunktion* (s. Abb. 5.5), *Fehleranzahl (s. Abb. 5.7)* und *Komplexität der Menüführung* (s. Abb. 5.6) angepasst werden.

```
//Parameter1: Name des Tasks
client.addHelpFunctionInvokation(task1);
```

Listing 5.5: Aufruf der Hilfefunktion hinzufügen

```
//Parameter1: Name des Tasks, Parameter2: aktueller View des
    //Menüelements, Parameter3: nachfolgender View
client.addMenuElement(taskName, actView, nextView)
```

Listing 5.6: Menü-Element für die Komplextität hinzufügen

```
//Exception hinzufügen
//Parameter1: Name des Tasks
MenuActivity.client.addException("allgemein");

//Fehler hinzufügen
//Parameter1: Name des Tasks
MenuActivity.client.addFailure("allgemein");
```

Listing 5.7: Fehler hinzufügen

Am Ende wird die Messung jeweils für die Tasks gestoppt. Der Benutzer bekommt einen Hinweis, ob die Daten zu dem jetzigen Zeitpunkt gestartet werden sollen. Bestätigt er dies, werden jene gesendet (s. Abb. 5.8).

```
Intent intentUsability = new Intent(this,
   UploadUserStatistic.class);
this.startService(intentUsability);
Toast.makeText(this, R.string.uploadUser_started, Toast.
   LENGTH_LONG).show();
```

Listing 5.8: Senden der Daten

Der Feedback-Fragebogen sollte nicht bei der ersten Bedienung der Anwendung abgefragt werden, weil sich der Benutzer noch keinen detaillierten Eindruck über die Anwendung machen konnte. Über eine Variable in der *SharedPreferences* kann die Anzahl der Starts festgehalten werden. Bei Erreichen dieser Zahl kann der Feedback-Fragebogen geladen werden (s. Abb. 5.9).

```
    //Abfrage über die Startversuche
    if (countApp == 10) {
    //Parameter1: Fragen Parameter2: Antworten
      client.loadUserFeedback(q, a);
    }
```

Listing 5.9: Feedbacken-Fragebogen laden

### 5.3.3 Probleme bei der Integration

Bei der Integration der Usability-Komponente in die Anwendung *Learn!* wurde festgestellt, dass die abgeleitete Klasse *UsabilityActivity* zur Messung der Touch- und Key Events nicht ausreicht. Es müssen weitere Oberklassen für die zusätzlich möglichen Activities z.B. *TabActivitiy* und *ListActivity* in der Android API entwickelt werden. In der Anwendung wurden *ListActivity* und *TabActivity* genutzt. Hierfür wurden zusätzliche Oberklassen entwickelt und in die Komponente *Usability* integriert (s. Abb. 5.8). Im Zuge der Weiterentwicklung von Android werden noch weitere Activities entstehen. Für jede dieser neuen Activities muss eine entsprechende Oberklasse entwickelt und in die Komponente *Usability* integriert werden.

Bei der Integration von fremden Programmbibliotheken können Probleme auftreten. Die Bibliotheken können auf Funktionen der Java Bibliothek zugreifen, die auf der angepassten Java-VM von Android nicht laufen oder zu Problemen führen. Bei XStream (s. Abschnitt 5.1.2.3) zum Beispiel funktioniert die Serialisierung mit Annotations nicht. Es gab hier aber die Möglichkeiten, jene ohne Annotations durchzuführen.

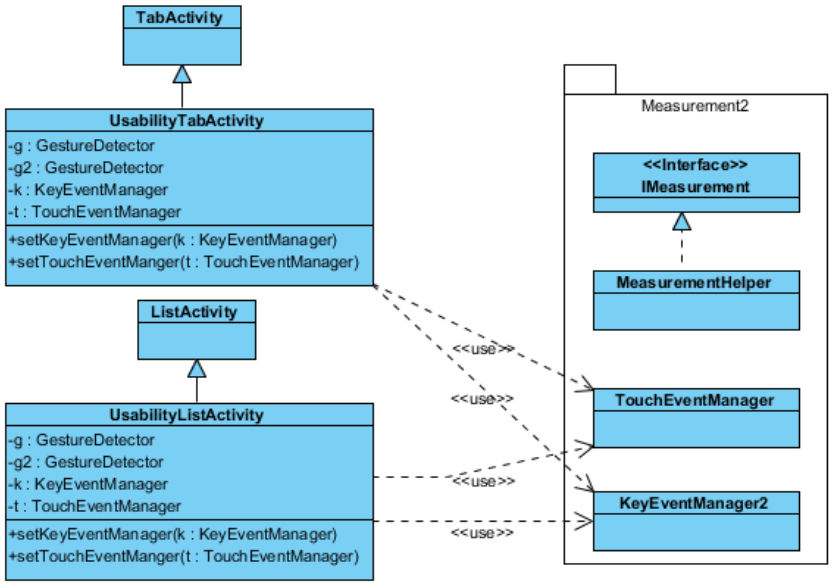



Abbildung 5.8: Diagramm der zusätzlichen Klassen für die Komponente *Measurement*

## 5.4 Testkonzept

Testen ist ein wichtiger Bestandteil bei der Entwicklung einer Anwendung. Das Testkonzept und das systematische Erstellen und Durchführen von Tests dient zur Qualitätssicherung

und zur Validierung der Anforderungen. Im Entwurf wurde die Usability-Komponente in fachliche Untereinheiten (Komponenten) zerlegt. Die Zugriffe erfolgen über deren Schnittstellen. Anschließend wurde die Usability-Komponente in das Anwendungsbeispiel integriert. Um die Anforderungen der einzelnen Komponenten zu testen, werden Komponententests durchgeführt. Das Zusammenspiel der einzelnen Komponenten erfordert ein Integrationstest. Abschließend erfolgt ein Systemtest am fertigen Endprodukt, also dem Anwendungsbeispiel mit integrierter Usability-Komponente. Beim Systemtest werden insbesondere die nicht-funktionalen Anforderungen getestet. Die unterschiedlichen Testarten erfordern so genannte Mock-Objekte (s. Abschnitt 5.4.2.2). Im Rahmen dieser Arbeit erfolgten die Tests auf dem Android-Emulator und den mobilen Endgeräten T-Mobile G1/HTC Magic.

### 5.4.1 Testaufbau

In diesem Abschnitt werden die unterschiedlichen Testarten beschrieben, die für einen vollständigen Test der Usability-Komponente und des Anwendungsbeispiels mit integrierter Usability-Komponente notwendig sind. Jede dieser Testarten besteht aus den Testphasen Planung, Vorbereitung, Durchführung und Auswertung. Es sollte versucht werden, die Tests automatisiert in den Buildprozess einzugliedern, da sie dann bei jedem Entwicklungsschritt im Buildprozess automatisch ausgeführt werden.

#### 5.4.1.1 Komponententest

Im Komponententest werden die einzelnen Komponenten auf korrekte Funktionalität getestet. Im ersten Schritt werden aus den Anforderungen der Komponenten Test-Szenarien entwickelt. Die Szenarien stellen sicher, dass der Aufruf der Schnittstellen entsprechend der Spezifikation abläuft. Dazu werden Funktions- und Schnittstellentests durchgeführt.

#### 5.4.1.2 Integrationstest

Mit dem Integrationstest wird das Zusammenspiel voneinander abhängiger Komponenten getestet. Die Test-Szenarien stellen sicher, dass beim Zusammenspiel der Komponenten der Zugriff über die Schnittstellen entsprechend der Spezifikation abläuft. Auch hier werden sowohl Funktionstests als auch Schnittstellentests vorgenommen. In diesem Fall sind die Untereinheiten unabhängig voneinander programmiert wurden. Nur für die Komponente *Usability* muss ein Integrationstest durchgeführt werden, weil nur sie auf die anderen unabhängigen Komponenten zugreift.

#### 5.4.1.3 Systemtest

Im Systemtest wird das gesamte System gegen die funktionalen und nicht-funktionalen Anforderungen getestet. Der Test sollte auf einem mobilen Endgerät mit echten Daten stattfinden. Der vollständige Systemtest muss beim Anwendungsbeispiel mit der integrierten Usability-Komponente erfolgen.

### 5.4.2 Durchgeführte Tests

An dieser Stelle werden die tatsächlichen durchgeführten Tests beschrieben. Außerdem werden weitere mögliche Tests genannt, die in einer zukünftigen Weiterentwicklung in Betracht kommen. Sie können jedoch im Rahmen dieser Arbeit nicht mehr geleistet werden.

#### 5.4.2.1 Entwicklung eines Prototyps

In der Realisierung wurde die Usability-Komponente als Prototyp umgesetzt. Die anschließende Integration in das Anwendungsbeispiel dient als Machbarkeitsnachweis. Außerdem wurde damit bewiesen, dass alle Komponenten innerhalb der Usability-Komponente sich nutzen lassen. Die Funktionalitäten der Komponenten wurde mit Mock-Objekten (s. Abschnitt 5.4.2.2) getestet.

#### 5.4.2.2 Mock

Mit Mock-Objekte können echte Daten simuliert werden. Sie stellen sicher, dass alle Methodenaufrufe vollständig und mit den korrekten Parametern gefüllt sind. Das Mock-Objekt liefert dabei ein Ergebnis zurück, welches auf seine Korrektheit hin überprüft werden kann. Bei der Entwicklung konnten die Komponenten und deren Schnittstellen im Rahmen dieser Arbeit mit Hilfe von Mock-Objekten getrennt getestet werden.

#### 5.4.2.3 Logging und Debugging

Um den umgesetzten Code und seinen Ablauf zu überprüfen, gibt es das Logging und Debugging im Android-Emulator. Auf diese Weise kann der Programmablauf überwacht werden. Um zu testen, ob der Code an der erwarteten Stelle ausgeführt wurde oder Exceptions geworfen wurden, können mit dem *Logging* Ausgaben erstellt werden.

#### 5.4.2.4 Integrationstest

Es wurden verschiedene Integrationstests für die Komponente *Usability* durchgeführt. Es wird exemplarisch anhand eines Szenarios ein Integrationstest der Komponente *Usability* vorgestellt.

**Planung** Für den Integrationstest werden die Komponenten *Usability* und für das Zusammenspiel notwendige unabhängige Komponenten genommen. Dort werden Funktions- und Schnittstellentests vorgenommen, um die Korrektheit der Spezifikation zu überprüfen.

**Vorbereitung** Exemplarisch wird die Anforderung *Konfiguration einlesen* (s. Kapitel 3.2.1.1) als Grundlage für den Test gewählt. Aus dem Sequenzdiagramm (s. Abb. 4.29) geht hervor, dass die Komponenten *Usability*, *Configuration*, *Measurement* zur Erfüllung dieser Anforderung notwendig sind.

**Testfall:**

1. Konfiguration (s. Anhang B.2) des Anwendungsbeispiels einlesen
   (Komponente: *Usability*, Schnittstelle: *Konfiguration einlesen* (s. Kapitel 4.2.1.9)), (Komponente: *Configuration*, Schnittstelle: *Konfiguration einlesen* (s. Kapitel 4.2.1.8))
2. Einlesen der Konfiguration prüfen
3. Messung erstellen
   (Komponente: *Measurement*, Schnittstelle: *Messung erstellen* (s. Kapitel 4.2.1.5))
4. Überprüfen, ob die entsprechenden Messungen in der Komponente *Measurement* vorhanden sind
5. Test erfolgreich, wenn 2. und 4. korrekt sind

**Durchführung** Mit Hilfe von *Logging und Debugging* (s. Abschnitt 5.4.2.3) und dem Emulator wurde der Testfall ausgeführt.

**Auswertung** Der Testfall wurde erfolgreich abgeschlossen.

### 5.4.3 Weitere Arten des Testens

Eine weitere Möglichkeit des Testens bietet das Testrahmenwerk JUnit. Mit dessen Hilfe können die Methoden auf Klassen- und Komponentenebene getestet werden. Im Rahmen dieser Arbeit wurde diese Art des Testens nicht umgesetzt. Bei einer späteren Weiterentwicklung sollte jene hinzugezogen werden, da sie das Testen automatisiert. In der Realisierung wurde ebenfalls kein kompletter Systemtest durchgeführt. Die funktionalen Anforderungen wurden überprüft, aber es wurden nicht alle nicht-funktionalen Anforderungen überprüft. Die nicht-funktionale Anforderung Effizienz (s. Kapitel 3.3.1) erfordert entsprechende Tests, um die Verzögerungszeiten durch die Usability-Komponente zu messen. Ebenfalls sind weitere Tests für die nicht-funktionale Anforderung Zuverlässigkeit (Robustheit) (s. Kapitel 3.3.2) notwendig. Zusätzlich fehlen entsprechende Tests für *Safety* (s. Kapitel 3.3.5.1) und *Security* (s. Kapitel 3.3.5.2). Für einen vollständigen Systemtest ist dies aber notwendig. Beim Senden der Daten wird über das mobile Endgerät eine Verbindung aufgebaut. Es ist aber nicht gewährleistet, dass das mobile Netz immer stabil eine Verbindung aufrecht erhält. Es können Schwankungen in der Übertragungsleistung bis hin zu kurzen Verbindungsabbrüchen auftreten. Diese müssen ebenfalls simuliert und getestet werden.

## 5.5 Fazit

In diesem Kapitel wurde der Softwareentwurf aus Kapitel 4 durch eine prototypische Realisierung umgesetzt. Die Realisierung erfüllt dabei alle beschriebenen funktionalen Anforderungen. Der Prototyp wurde in ein Anwendungsbeispiel integriert. Diese wurde auf dem Emulator und den beiden zur Verfügung stehenden mobilen Endgeräten ausgeführt und zusammen mit einer beispielhaften Implementierung eines lokalen Usability-Servers getestet.

# 6 Bewertung

In diesem Kapitel wird das Design aus dem Kapitel 4 und die Umsetzung aus Kapitel 5 kritisch bewertet. Dabei wird beschrieben, welche funktionalen und nicht-funktionalen Anforderungen aus dem Kapitel 3 in dem Design und der Realisierung erfüllt werden konnte. Außerdem sollen die aufgetretenen Mängel beschrieben und Lösungsansätze aufgezeigt werden.

## 6.1 Anforderungsabgleich

In diesem Abschnitt werden die funktionalen und nicht-funktionalen Anforderungen aus dem Kapitel 3 mit dem Design und der Realisierung verglichen. Mit dessen Hilfe können Lücken im Design oder in der Realisierung aufgezeigt werden.

### 6.1.1 Funktionale Anforderungen

Die nachfolgenden Tabelle 6.1 zählt die funktionalen Anforderungen aus Kapitel 3.2 auf und die Spalten Design und Realisierung zeigen, inwiefern diese dort umgesetzt wurden.

Das Usecase Diagramm (s. Abb. 3.5) beinhaltet die Akteure *Mobile Anwendung*, *Serveranwendung*, *Entwickler* und *Usability-Experte*. Die Anwendungsfälle *Task definieren* (s. Kapitel 3.2.1.1), *Messgrößen festlegen* (s. Kapitel 3.2.1.1) und *Auswertung der Daten einsehen* (s. Kapitel 3.2.1.5), in denen der Usability-Experte beteiligt ist, wurden im Design und der Realisierung nicht weiter betrachtet. Das Definieren der Tasks und Messgrößen wird in der Regel vor der eigentlichen Integration der Usability-Komponente durchgeführt. Die Auswertung der Daten hat mit der Entwicklung der Usability-Komponente nicht direkt zu tun. Dieser Schritt wird nach der Integration durchgeführt.

Die Anwendungsfälle Completion Rate (s. Kapitel 3.2.1.2) und Komplexität der Menüführung (s. Kapitel 3.2.1.2) wurden umgesetzt. Leider muss der Entwickler für diese Messgrößen manuell in seiner Anwendung Code implementieren.

| Anforderungen | Design | Realisierung |
|---|---|---|
| Task definieren (s. Kapitel 3.2.1.1) | - | - |
| Task konfigurieren (s. Kapitel 3.2.1.1) | X | X |
| Messgrößen festlegen (s. Kapitel 3.2.1.1) | - | - |
| Messgrößen konfigurieren (s. Kapitel 3.2.1.1) | - | - |
| Konfiguration einlesen (s. Kapitel 3.2.1.1) | X | X |
| Zeitaufwand für die Aufgabe (Task Time) (s. Kapitel 3.2.1.2) | X | X |
| Completion Rate (s. Kapitel 3.2.1.2) | X | X |
| Fehleranzahl (s. Kapitel 3.2.1.2) | X | X |
| Aufruf der Hilfefunktion (s. Kapitel 3.2.1.2) | X | X |
| Anzahl der Touch Events (s. Kapitel 3.2.1.2) | X | X |
| Anzahl der Key Events (s. Kapitel 3.2.1.2) | X | X |
| Komplexität der Menüführung (s. Kapitel 3.2.1.2) | X | X |
| Protokollierung (s. Kapitel 3.2.1.3) | X | X |
| Benutzerinteraktion (s. Kapitel 3.2.1.4) | X | X |
| Auswertung der Daten (s. Kapitel 3.2.1.5) | X | X |
| Auswertung der Daten einsehen (s. Kapitel 3.2.1.5) | - | - |
| Senden der Daten (s. Kapitel 3.2.1.6) | X | X |
| Empfang der Daten (s. Kapitel 3.2.1.6) | X | X |

Tabelle 6.1: Umsetzung der funktionalen Anforderungen

### 6.1.2 Nicht-funktionale Anforderungen

In Tabelle 6.2 werden die nicht-funktionalen Anforderungen aufgeführt und die Umsetzung mit dem Design und der Realisierung abgeglichen.

Durch den begrenzten Zeitrahmen der Arbeit wurden nicht alle nicht-funktionalen Anforderungen umgesetzt. Die Sicherheit wurde für diese Arbeit ausgeklammert und nicht weiter betrachtet. Die anderen nicht-funktionalen Anforderungen wurden im Design berücksichtigt. In der Realisierung wurden sie nur zum Teil einbezogen. In den Komponenten wurden Fehlerverursacher durch entsprechende Exceptions abgefangen. Die Kommunikation zwischen mobilen Endgerät und dem Usability-Server wurde aber nicht weiter getestet. Für die Effizienz wurde im Design versucht, die Hardware-Ressourcen zu schonen. Es fehlen aber entsprechende Tests in der Realisierung, um die Grenzen der Verzörgungszeit zu bestätigen. Ebenso fehlen systematische Test in der Realisierung für *Wartbarkeit (Verständlichkeit)* (s. Kapitel 3.3.6) und *Benutzbarkeit (Erlernbarkeit)* (s. Kapitel 3.3.4).

| Anforderungen | Design | Realisierung |
|---|---|---|
| Effizienz (s. Kapitel 3.3.1) | X | (X) |
| Zuverlässigkeit (Robustheit) (s. Kapitel 3.3.2) | X | (X) |
| Zuverlässigkeit (Korrektheit) (s. Kapitel 3.3.3) | X | (X) |
| Benutzbarkeit (Erlernbarkeit) (s. Kapitel 3.3.4) | X | (X) |
| Safety (s. Kapitel 3.3.5.1) | - | - |
| Security (s. Kapitel 3.3.5.2) | - | - |
| Wartbarkeit (Verständlichkeit) (s. Kapitel 3.3.6) | X | (X) |

Tabelle 6.2: Umsetzung der nicht-funktionalen Anforderungen

## 6.2 Durchführung

Die Usability-Komponente wurde in der Realisierung in die Applikation Learn! integriert. Anschließend wurde die Anwendung auf dem Emulator und den mobilen Endgeräten ausgeführt. Während der Ausführung wurden die Szenarien aus Kapitel 5.2 ausgeführt. Anschließend wurden die Ergebnisse an einen lokalen Usability-Server im LAN gesendet. Die Auswertung fand dann auf dem Server statt. Das Ergebnis der Auswertung wurde in ein Excel-Dokument geschrieben. In den weiteren Abschnitten werden die Erkenntnisse aus der Usability-Untersuchung dargestellt und abschließend bewertet.

### 6.2.1 Erkenntnisse ohne Usability-Komponente

Die folgenden Erkenntnisse wurden bei der Ausführung der Szenarien ohne die Usability-Komponente erworben. Sie sind nicht objektiv, sondern entsprechen der subjektiven Wahrnehmung des Autors. Zusätzlich wurden die Guidelines (s. Kapitel 2.5) zur Überprüfung der Anwendung zur Hilfe genommen.

Beim Ausführen einer Lerneinheit aus Szenario 2 (s. Kapitel 5.2) kann über das Menü die Lerneinheit nicht beendet werden, nur über die „Zurück“-Taste ist ein Beenden möglich. Die „Zurück“-Taste für die Navigation sollte mit Vorsicht genossen werden, wozu der Guideline von Google [Alliance (2010)] rät. Komplexe Navigation kann zu Verwirrungen beim Benutzer führen. Wichtige Navigationspunkte in einer Anwendung sollten über das „Options“-Menü anwählbar sein. Dazu gehört auch das Ende der Lerneinheit.

Im Ablauf des Szenario 2 (s. Kapitel 5.2.1) kann man eine Lerneinheit auswählen. Die Activity aus dem Szenario 2 gleicht dem aus Szenario 1 ohne den Eintrag *Add new Collection*. Bei gleich aussehenden Activities könnte der Benutzer die gleichen Funktionalitäten erwarten. Im Gegensatz zu Szenario 1 fehlt aber in Szenario 2 der Menüeintrag, um neue Collections zu erstellen.

Es gibt in der Anwendung nur die Möglichkeit, die Hilfe über das Hauptmenü aufzurufen. Während der Ausführung der Anwendung muss zum Hauptmenü navigiert werden, um die Hilfe aufzurufen. Es wäre wünschenswert, die Hilfe in ihrem Kontext während der Bedienung aufzurufen. Hierzu raten auch die Guidelines [Nokia (2010), Wiki (2010)]. Die Hilfefunktion kann in der Anwendung auch nur online über den Browser abgerufen werden; sie sollte aber auch ohne eine Internetverbindung abrufbar sein.

### 6.2.2 Erkenntnisse mit Usability-Komponente

Mit der Usability-Komponente lassen sich die Usability-Attribute (s. Kapitel 2.1.2) anhand der Messgrößen (s. Kapitel 3.2.1.2) überprüfen. Über die Konfiguration (s. Anhang B.4) wurden die Messgrößen festgelegt. *Exception* und *FailureRate* gehören zur Messgröße *Fehleranzahl*. Hiermit lassen sich die Anzahl von Fehlern und Exception in der Anwendung bestimmen. Die *Fehleranzahl* hat Einfluss auf das Usability-Attribut *Fehlertoleranz* (s. Kapitel 2.1.2). In der Anwendung wurden bei kritischen Exceptions und Fehlern mit der Hilfe der Usability-Komponente diese zusammen mit dem Logging vermerkt. Hiermit lassen sich grobe Programmfehler aufdecken. Es traten während der Ausführung der Szenarien keine Fehler/Exceptions auf. Die Messgröße *TaskTime* zeigt die Ausführungszeit der Tasks an. Eine hohe Ausführungszeit kann auf Probleme während der Ausführung der Tasks hinweisen. Sie beeinflusst nämlich das Usability-Attribut *Effizienz* (s. Kapitel 2.1.2). Die Ausführung der Tasks

ergaben keine hohen Ausführungszeiten. Für eine genauere Aussage fehlen weitere Datensätze von mehreren Benutzern. Für den Vergleich fehlen zusätzlich für die entsprechenden Tasks Referenzwerte, die in einer Vorab-Untersuchung vom Entwickler/Usability-Experten festgelegt werden muss. Die quantitativen Ausprägungen der Messgröße *Anzahl der Touch Events* oder *Anzahl der Key Events* können ebenfalls ein Hinweis auf Probleme in der Bedienung geben. Die beiden Messgrößen haben Einfluss auf das Usability-Attribut *Effizienz* (s. Kapitel 2.1.2). Die quantitativen Ausprägungen zeigten aber keine hohen Werte an. Die quantitativen Ausprägungen Tiefe und Breite der Messgröße *Komplexität der Menüführung* haben Auswirkungen auf die Usability-Attribute *Effektivität* (s. Kapitel 2.1.2) und *Lernbarkeit* (s. Kapitel 2.1.2). Die beiden können auf eine weit verzweigte Anwendung hinweisen. Hohe Werte dieser beiden Ausprägungen erschweren die *Lernbarkeit* der Anwendung und vor allem hat es Auswirkungen auf die *Effektivität*. Die quantitativen Ausprägungen Tiefe und Breite der Messgröße *Komplexität der Menüführung* hatten geringe Werte, welches auf eine flache und nicht tiefe Menüstruktur hinweist. Aufgrund der mangelnden Hilfe-Möglichkeiten können über die Messgröße *Aufruf der Hilfefunktion* nicht direkt Erkenntnisse geschlossen werden. Die eingeschränkte Hilfe-Möglichkeit wurde in Abschnitt 6.2.1 erwähnt. Auf den Feedback-Fragebogen wurde während der Ausführung verzichtet, da der Autor diesen hätte erstellen und gleichzeitig ausfüllen müssen.

### 6.2.3 Bewertung der Durchführung

Die Implementierung der Usability-Komponente in das Anwendungsbeispiel *Learn!* erwies sich als komplex. Dem Autor standen nur eine kleine Dokumentation und der Sourcecode der Anwendung zur Verfügung. Das Einarbeiten ohne ausführliche Dokumentation und UML-Diagramme machte es schwierig die Anwendung und ihren Code vollständig zu erfassen. Es ist durchaus möglich, dass beim Einbinden der Usability-Komponente nicht alle relevanten Bestandteile der Tasks im Code erfasst wurden. Die Integration der Usability-Komponente war dennoch tiefgreifend genug, so dass eine umfassende Bewertung der Ergebnisse möglich ist.

### 6.2.4 Bewertung der Ergebnisse

Die Usability-Untersuchung des Anwendungsbeispiels macht deutlich, dass nicht nur über die quantitativen Ausprägungen der Messgrößen Erkenntnisse über Usability gewonnen werden können. Ebenfalls lassen sich Usability-Probleme durch Benutzerbefragung erfassen. Für ein vollständiges Erfassen der Usability-Probleme sind Feld- und Laboruntersuchungen (s. Kapitel 2.2) notwendig. Aus dem Kapitel wird deutlich, dass Interviews mit dem Benutzer weitere Erkenntnisse über die Usability liefern können. In der Usability-Komponente wurde

mit dem Feedback-Fragebogen versucht, die subjektive Wahrnehmung des Benutzers zu erfassen. Mit dem Fragebogen hat man nur begrenzte Möglichkeiten, Probleme in der Anwendung zu erfassen. Es lassen sich nur allgemeine Probleme erfassen. Der Benutzer kann keine eigenen Anmerkungen oder Hinweise auf Schwierigkeiten geben. Im Gegensatz zu klassischen Labor-Untersuchungen fehlt ein Dialoggespräch zwischen dem Experten und dem Benutzer. Die Usability-Komponente kann erst voll zur Geltung kommen, wenn viele Benutzer ihre Ergebnisse an den Usability-Server übertragen haben und sie dort ausgewertet werden. Sie steigern damit die Anzahl der Datensätze und damit die Aussagekraft der Statistik. Außerdem muss festgehalten werden, dass man durch eine Weiterentwicklung der Usability-Komponente (insbesondere der Messgrößen) die Aussagequalität der Untersuchungen noch weiter steigern kann. Sie wird aber nie eine klassische Usability-Untersuchung im Labor ersetzen können. Schon gar nicht können mit der Usability-Komponente die gleichen Usability-Probleme erkannt werden, wie mit einer Labor-Untersuchung. Dies zeigte auch der Vergleich zwischen einer Labor- und Feldstudie in Kapitel 2.2.3. Sie kann als solche aber die Untersuchung im Labor ergänzen. Im Kapitel 2.5 wurden mit Hilfe einer Tabelle vorhandene Studien und Entwicklungen im Bereich Usability und Android in Kategorien unterteilt. Die Usability-Komponente ordnet sich in die Tabelle 6.3 folgendermaßen ein:

| | Benutzer-interaktion | Usability-Attribute | PC Appl. | Mobile Appl. | Benutzer-. befragung | Messdaten sammeln/ auswerten | Labor | Feld |
|---|---|---|---|---|---|---|---|---|
| Usability -Komp. | | X | | X | | X | X | X |

Tabelle 6.3: Einordnung der Arbeit

## 6.3 Verbesserung

Bei der Entwicklung eines Prototypen entstehen zwangsläufig Verbesserungsvorschläge für eine Weiterentwicklung der Software. Der Abschnitt beschreibt zukünftige Verbesserungsmöglichkeiten der Usability-Komponente.

### 6.3.1 Messgrößen

Die Bewertung der Ergebnisse zeigte, dass Usability nur begrenzt durch die jetzigen Messgrößen erfassbar ist. In einem weiteren Schritt sollte untersucht werden, welche zusätzlichen

Messgrößen die Aussagequalität von Usability vergrößert. Es wäre wünschenswert, die jetzigen Messgrößen *Komplexität der Menüführung*, *Completion Rate* und zukünftigen Messgrößen weiter zu automatisieren, um das Einbinden in den Sourcecode der Anwendung zu erleichtern und vor allem den Aufwand zu minimieren.

### 6.3.2 Usability-Server

In der prototypischen Realisierung spielt der Usability-Server nur eine untergeordnete Rolle. Die Auswertung wird mit Hilfe einer Exceldatei erstellt. In einem weiteren Schritt sollte ein vollständiger Usability-Server entwickelt werden. Auf Basis einer Web-Applikation können die Daten empfangen, entsprechend ausgewertet und über einen Browser abgerufen werden. In dem Usecase-Diagramm (s. Abb. 3.5) wurden verschiedenen Akteure erwähnt. Hier könnte die Web-Applikation für den jeweiligen Akteur verschiedene Sichten auf die Ergebnisse und die Weiterentwicklung geben.

### 6.3.3 Test

In der prototypischen Realisierung der Usability-Komponente wurden auf JUnit-Tests verzichtet. In einer Weiterentwicklung der Usability-Komponente und Entwicklung eines Usability-Servers sollten JUnit-Tests eingebaut werden, um automatisiert die Komponenten und deren Zusammenspiel zu testen. Ein wichtiger Punkt ist auch das Testen der nicht-funktionalen Anforderungen, um einen vollständigen Systemtest durchführen zu können. Diese werden in den weiteren Abschnitten im Detail beschrieben.

#### 6.3.3.1 Kommunikation

Die Usability-Komponente baut über eine drahtlose Verbindung WLAN/UMTS/GPRS eine Verbindung zum Usability-Server auf. Im Zuge eine Weiterentwicklung muss diese Kommunikation getestet werden. Insbesondere das Verhalten bei Verbindungsproblemen oder Verbindungsabbrüchen. Hiermit wird die nicht-funktionale Anforderung *Zuverlässigkeit (Robustheit)* (s. Kapitel 3.3.2) geprüft.

#### 6.3.3.2 Performance

Eine weitere nicht-funktionale Anforderung ist die *Effizienz* (s. Kapitel 3.3.1). Es wurde festgelegt, dass durch die Integration der Usability-Komponente die maximale Verzögerungszeit

bei 500ms liegen sollte. Hierfür müssen systematische Tests auf den beiden mobilen Endgeräten erfolgen.

### 6.3.4 Sicherheit

Im Bereich Sicherheit wurden keine Tests geplant und durchgeführt. Die Sicherheit beinhaltet die nicht-funktionalen Anforderungen *Safety* (s. Kapitel 3.3.5.1) und *Security* (s. Kapitel 3.3.5.2), welche im Zuge einer Weiterentwicklung getestet werden müssen.

### 6.3.5 Integration

Die Usability-Komponente wird in bestehende oder noch zu entwickelnde Anwendungen integriert. Die Integration sollte dem Anwendungsentwickler erleichtert werden. Hierfür stehen die nicht-funktionalen Anforderungen *Benutzerbarkeit (Erlernbarkeit)* (s. Kapitel 3.3.4) und *Wartbarkeit (Verständlichkeit)* (s. Kapitel 3.3.6). In einem Beta-Test können diese Anforderungen getestet werden, indem ein Dialog mit den Anwendungsentwicklern erfolgt. Mit Hilfe der Feedbacks können die Probleme erfasst und als Grundlage für die Weiterentwicklung dienen.

## 6.4 Fazit

In diesem Kapitel wurden die funktionalen und nicht-funktionalen Anforderungen mit dem Design und der Realisierung abgeglichen. Es zeigte sich, dass die relevanten funktionalen Anforderungen für die prototypische Umsetzung im Design und der Realisierung komplett umgesetzt wurden. Die nicht-funktionalen Anforderungen dagegen wurden nur zum Teil umgesetzt. Anschließend wurde die Durchführung beschrieben und deren Ergebnisse bewertet. Zum Schluss wurden auf mögliche Verbesserungen in einer zukünftigen Entwicklung hingewiesen.

# 7 Zusammenfassung und Ausblick

Im letzten Kapitel wird die Arbeit noch einmal zusammengefasst und ein Ausblick auf zukünftige Weiterentwicklungen gegeben.

## 7.1 Zusammenfassung

Im Rahmen dieser Arbeit wurde eine Usability-Komponente für Android-Applikationen entworfen und prototypisch realisiert. Mit Hilfe der Usability-Komponente haben die Entwickler die Möglichkeit, Usability-Rückschlüsse auf deren Applikation zu erhalten. Die klassischen Usability-Untersuchungen werden in einem Labor ausgeführt. Sie sind zum Teil aufwendig und kostspielig. Sie spiegeln auch nur teilweise den natürlichen Kontext der Anwendung wieder. Die Zielgruppe sind die kleinen Entwicklerteams. Mit der Usability-Komponente steht ihnen eine kostengünstige Möglichkeit zur Verfügung, um eine Usability-Untersuchung im natürlichen Kontext durchzuführen. Zusätzlich kann es eine klassische Laboruntersuchung ergänzen.

Zu Beginn der Arbeit (s. Kapitel 2) wurden die grundlegenden Themenbereiche *Usability* und *Android* betrachtet, insbesondere die Herausforderung, Usability auf mobilen Endgeräten zu messen. Außerdem wurde hier der Unterschied zwischen Labor- und Felduntersuchung beschrieben und diskutiert. Es stellte sich heraus, dass beide Untersuchungsarten verschiedene Usability-Probleme erfassen können.

In Kapitel 3 wurden zur Hilfenahme eines Anwendungsbeispiels die funkionalen und nichtfunktionalen Anforderungen an die Usability-Komponente erfasst und für einen Entwurf spezifiziert. Auf dessen Basis wurde in Kapitel 4 ein Prototyp der Usability-Komponente entwickelt. Zusätzlich zum Entwurf der Usability-Komponente wurde jene auch für das mobile Endgerät und den Usability-Server vorgestellt, welche den Einsatz der Usability-Komponente auf beiden zeigt.

Für den Entwurf wurde eine komponentenbasierte Architektur gewählt. Über die Schnittstellen können die Komponenten kommunizieren. Mit diesem Ansatz werden die Funktionalitäten der Usability-Komponente gekapselt. Außerdem erhöht der gewählte Ansatz die

Verständlichkeit, Änderbarkeit und Erweiterbarkeit. In der Realisierung (s. Kapitel 5) wurde der Entwurf prototypisch mit den aktuellen Technologien umgesetzt. Der Usability-Server wurde nur eingeschränkt umgesetzt. Anschließend wurde mit Hilfe eines Anwendungsbeispiels die Integration beschrieben und durchgeführt. Die funktionalen Anforderungen aus Kapitel 3 bildeten die Grundlagen des Testkonzepts (s. Kapitel 5.4), womit die Funktionalitäten des Entwurfes überprüft wurden. Die Bewertung machte deutlich, dass die geforderten Anforderungen zum größten Teil erfüllt wurden. Gleichzeitig ergaben sich aber auch weitere Verbesserungsvorschläge für zukünftige Weiterentwicklungen.

## 7.2 Ausblick

In der Arbeit wurde ein Prototyp von der ersten Idee bis hin zur Umsetzung realisiert. Die Umsetzung machte deutlich, dass für eine Bewertung der Usability weitere Messgrößen erforderlich sind. Hierfür sind zusätzliche Untersuchungen notwendig, um weitere Messgrößen zu entwerfen. Aus der Bewertung ging hervor, dass Usability nicht nur durch Messgrößen erfasst werden kann. Durch die subjektive Wahrnehmung können weitere Usability-Probleme erfasst werden. In zukünftigen Untersuchungen muss festgestellt werden, inwiefern die subjektive Wahrnehmung messbar gemacht werden kann. In der Arbeit wurde der Usability-Server nur rudimentär beschrieben und entwickelt. In einer Weiterentwicklung muss ein vollständiger Usability-Server z.B. als Web-Applikation entworfen und realisiert werden. Abschließend ist es zwingend notwendig, weitere Tests zu entwerfen und automatisiert durchzuführen, um das System vollständig prüfen zu können.

Zu guter Letzt sollte die Usability-Komponente über eine Plattform wie Sourceforge [Sourceforge (2010)] veröffentlicht und bereitgestellt werden. Die Verbreitung der Usability-Komponente sollte über einschlägige Entwicklerforen erfolgen. Hierdurch kann ein Dialog mit den Entwicklern der Android-Applikationen stattfinden. Somit lassen sich die Probleme im Umgang mit der Usability-Komponente erfassen, und zusammen können sie mit den Verbesserungsvorschläge in die Weiterentwicklung einfliessen.

# A Anwendungsfälle

## A.1 Konfiguration

### A.1.1 Task definieren

**Titel:** **Task definieren**
**Akteur:** **Usability-Experte**
**Ziel:** **Task definieren**

**Auslöser:**
Der Usability-Experte bekommt die Aufgabe für eine bestehende Anwendung die Tasks herauszufiltern, die für die Usability-Untersuchung der mobilen Anwendung notwendig sind.

**Vorbedingungen:**
Eine bestehende oder noch zu entwickelnde Anwendung für ein mobiles Endgerät.

**Nachbedingungen:**
Task muss vollständig erfasst sein.

**Erfolgsszenario:**

1. Usability-Experte testet die bestehende mobile Anwendung
2. Task werden definiert
3. Use Case „Messgrößen festlegen“

**Alternative:**

1. Usability-Experte testet einen Prototypen, der noch in der Entwicklung ist.

**Häufigkeit:**
Jeder Task, der für Usability in Frage kommt, muss vollständig definiert werden.

### A.1.2 Task konfigurieren

**Titel:** **Task konfigurieren**
**Akteur:** **Entwickler**
**Ziel:** **Konfiguration der Usability-Komponente**

**Auslöser:**
Der Usability-Experte kann vorab die Task definieren, welche für eine Usability-Untersuchung notwendig sind. Daraufhin integriert der Entwickler die Usability-Komponente in eine bestehende oder zukünftige Anwendung. Die Komponente muss dafür entsprechend konfiguriert werden.

**Vorbedingungen:**
Eine bestehende oder noch zu entwickelnde Anwendung für ein mobiles Endgerät.

**Nachbedingungen:**
Der Task wird vollständig in der Konfiguration definiert.

**Erfolgsszenario:**

1. Einbinden der Usability-Komponente in die Anwendung
2. Task in der Konfiguration definieren
3. Use Case „Messgrößen konfigurieren“

**Fehlerfälle:**

1. Task unvollständig definiert. Beim Einlesen der Konfiguration meldet die Usability-Komponente einen Fehler.

**Häufigkeit:**
Jeder Task, der für die Usability-Untersuchung in Frage kommt, muss in die Konfiguration eingetragen werden.

**Anforderungen:**

1. Benutzbarkeit (Erlernbarkeit):
   Die Konfiguration eines Tasks soll einfach gehalten werden. Eine entsprechende Dokumentation mit Beispielen soll dem Entwickler die Konfiguration erleichtern.
2. Wartbarkeit (Verständlichkeit):
   Nach kurzer Einarbeitung soll die Anpassung oder das Anlegen von Tasks für den Entwickler leicht durchführbar sein.

### A.1.3 Messgrößen festlegen

**Titel:** **Messgrößen festlegen**
**Akteur:** **Usability-Experte**
**Ziel:** **Relevante Messgrößen für die jeweiligen Tasks festlegen**

**Auslöser:**
Der Usability-Experte definiert einen Task.

**Vorbedingungen:**
Keine

**Nachbedingungen:**
Messgrößen sind vollständig definiert.

1. Messgrößen innerhalb des Tasks festlegen.

**Häufigkeit:**
Für jeden Task müssen mindestens ein oder mehr Messgrößen festgelegt werden.

### A.1.4 Messgrößen konfigurieren

**Titel:** **Messgrößen konfigurieren**
**Akteur:** **Entwickler**
**Ziel:** **Messgrößen für die Tasks konfigurieren**

**Auslöser:**
Der Entwickler bekommt vom Usability-Experten die relevanten Messgrößen vorgelegt. Anhand dessen legt der Entwickler die Messgrößen in der Konfiguration an.

**Vorbedingungen:**
Keine

**Nachbedingungen:**
Die Messgrößen sind vollständig konfiguriert. Im Sourcecode werden die Messgrößen korrekt benutzt.

**Erfolgsszenario:**

1. Messgrößen werden vollständig konfiguriert

2. Die notwendigen Anpassungen an den Sourcecode für die Messung werden vorgenommen

**Fehlerfälle:**

1. Konfiguration der Messgrößen unvollständig oder fehlerhaft. Beim Einlesen der Konfiguration meldet die Usability-Komponente einen Fehler.

**Häufigkeit:**
Jede eingetragende Messgröße muss entsprechend konfiguriert werden.

**Anforderungen:**

1. Benutzbarkeit (Erlernbarkeit):
   Die Konfiguration der Messgrößen sollte einfach gehalten werden. Die Dokumentation mit entsprechenden Beispielen soll dem Entwickler die Konfiguration erleichtern.
2. Wartbarkeit (Verständlichkeit):
   Nach kurzer Einarbeitung soll die Anpassung oder das Anlegen von Messgrößen für den Entwickler leicht durchführbar sein.

### A.1.5 Konfiguration einlesen

**Titel:** **Konfiguration einlesen**
**Akteur:** **Mobile Anwendung**
**Ziel:** **Messgrößen und Tasks aus der Konfiguration einlesen**

**Auslöser:**
Der Benutzer startet den ersten Task in der Anwendung.

**Vorbedingungen:**
Die Messgrößen und Tasks sind vollständig in der Konfiguration definiert. Außerdem hat der Benutzer der Usability-Untersuchung zugestimmt.

**Nachbedingungen:**
Das Einlesen der Konfiguration war erfolgreich.

**Erfolgsszenario:**

1. Benutzer beginnt mit dem ersten Task
2. Konfiguration wird eingelesen

**Fehlerfälle:**

1. Das Einlesen der Konfiguration bricht mit einem Fehler ab.

**Häufigkeit:**
Die Konfiguration wird nur beim ersten Start eines Tasks eingelesen.

**Anforderungen:**

1. Effizienz:
   Das Einlesen der Konfguration kann die Anwendung verzögern. Die maximale Verzögerungszeit sollte den Wert 500ms nicht überschreitrn.

## A.2 Messen

### A.2.1 Zeitaufwand für die Aufgabe

**Titel:** **Task Time**
**Akteur:** **Mobile Anwendung**
**Ziel:** **Zeitaufwand für den Task messen**

**Auslöser:**
Der Benutzer startet den Task.

**Vorbedingungen:**
Die Messgröße *Task Time* ist vollständig in der Konfiguration für den Task definiert. Außerdem hat der Benutzer der Usability-Untersuchung zugestimmt.

**Nachbedingungen:**
Der Task wurde erfolgreich beendet.

**Erfolgsszenario:**

1. Benutzer beginnt mit dem Task
2. Beim Start des Tasks wird der Anfangswert (Timestamp) mit dem Tasknamen gespeichert
3. Benutzer beendet den Task erfolgreich
4. Der Wert der spezifischen Messgröße wird zusammen mit dem zugehörigen Task auf dem mobilen Endgerät gespeichert

**Erweiterung:**

1. Der Benutzer bricht die Aktion ab. Dies führt zum Abbruch der Messung der spezifischen Messgröße.
2. Der Benutzer wird durch ein Telefongespräch, SMS oder anderen Notifications des mobilen Endgerätes unterbrochen. Die Zeit wird hierbei gestoppt bzw. die Unterbrechungszeit wird vom Zeitaufwand abgezogen.

**Fehlerfälle:**

1. Der Benutzer beendet die Anwendung bevor die Aufgabe erfolgreich abgeschlossen wurde. Dies führt zum Abbruch der Messung.
2. Der Absturz der Anwendung führt ebenfalls zum Abbruch der Messung.

**Häufigkeit:**
Pro Task kann einmal die Task Time bestimmt werden.

**Anforderungen:**

1. Effizienz:
   Die Usability-Komponente sorgt für eine größere Verzögerungszeit. Dies hat Einfluss auf die gesamte Ausführungszeit der Anwendung. Die Verzögerungszeit sollte maximal 5% des Zeitaufwandes betragen. Die Speicherung der Messgrößen auf dem mobilen Endgerät dürfte die Anwendung maximal um 250ms verzögern.
2. Sicherheit (Safety):
   Der Zeitaufwand darf nur beim erfolgreichem Beenden des Tasks gespeichert werden. Bei Absturz oder im Fehlerfall wird es keinen Endzeitpunkt für den Task geben. Ohne den gültigen Endzeitpunkt darf die Messgröße nicht gespeichert werden.

### A.2.2 Completion Rate

**Titel:** **Completion Rate**
**Akteur:** **Mobile Anwendung**
**Ziel:** **Effektivität beim erfolgreichem Ausführen des Task messen**

**Auslöser:**
Der Benutzer startet den Task.

**Vorbedingungen:**
Die Messgröße *Completion Rate* ist vollständig in der Konfiguration für den Task definiert. Außerdem hat der Benutzer der Usability-Untersuchung zugestimmt.

**Nachbedingungen:**
Der Task wurde erfolgreich beendet.

**Erfolgsszenario:**

1. Benutzer beginnt mit dem Task
2. Nach dem Starten des Tasks werden die Zwischenschritte für die spätere Auswertung protokolliert.
3. Benutzer beendet den Task erfolgreich
4. Der Wert der spezifischen Messgröße wird zusammen mit dem zugehörigen Task auf dem mobilen Endgerät gespeichert

**Fehlerfälle:**

1. Der Benutzer beendet die Anwendung bevor die Aufgabe erfolgreich abgeschlossen wurde. Dies führt zum Abbruch der Messung.
2. Der Absturz der Anwendung führt ebenfalls zum Abbruch der Messung.

**Häufigkeit:**
Pro Task kann einmal die Completion Rate bestimmt werden.

**Erweiterung:**

1. Der Benutzer bricht die Aktion ab. Dies führt zum Abbruch der Messung der spezifischen Messgröße.

**Anforderungen:**

1. Effizienz:
   Die Speicherung der spezifischen Messgröße oder die Protokollierung dürfen die Anwendung maximal um 250ms verzögern.

### A.2.3 Fehleranzahl

**Titel:** **Fehleranzahl**
**Akteur:** **Mobile Anwendung**
**Ziel:** **Messen der Fehler/ Ausnahmen (Exceptions) in der Anwendung**

**Auslöser:**
Der Benutzer löst einen Fehler/ Ausnahme (Exception) aus.

**Vorbedingungen:**
Die Messgröße *Fehleranzahl* ist vollständig in der Konfiguration für den Task vollständig. Außerdem hat der Benutzer der Usability-Untersuchung zugestimmt.

**Nachbedingungen:**
Der Task wurde erfolgreich beendet.

**Erfolgsszenario:**

1. Benutzer beginnt mit dem Task
2. Benutzer beendet den Task erfolgreich
3. Der Wert der spezifischen Messgröße wird zusammen mit dem zugehörigen Task auf dem mobilen Endgerät gespeichert

**Fehlerfälle:**

1. Der Benutzer beendet die Anwendung bevor die Aufgabe erfolgreich abgeschlossen wurde. Dies führt zum Abbruch der Messung.
2. Der Absturz der Anwendung führt ebenfalls zum Abbruch der Messung.

**Häufigkeit:**
Pro Task können für die Messgröße Fehleranzahl Exceptions und/oder Failurerate konfiguriert werden.

**Anforderungen:**

1. Effizienz:
   Die Speicherung der spezifischen Messgröße in die Datenbank oder die Protokollierung dürfen die Anwendung maximal um 250ms verzögern.

### A.2.4 Aufruf der Hilfefunktion

**Titel:** **Aufruf der Hilfefunktion**
**Akteur:** **Mobile Anwendung**
**Ziel:** **Anzahl der Hilfefunktionsaufrufe messen**

**Auslöser:**
Der Benutzer ruft die Hilfefunktion in der Anwendung auf.

**Vorbedingungen:**
Die Messgröße *Aufruf der Hilfefunktion* ist vollständig in der Konfiguration für den Task definiert. Außerdem hat der Benutzer der Usability-Untersuchung zugestimmt.

**Nachbedingungen:**
Der Task wurde erfolgreich beendet.

**Erfolgsszenario:**

1. Benutzer beginnt mit dem Task
2. Benutzer beendet den Task erfolgreich
3. Der Wert der spezifischen Messgröße wird zusammen mit dem zugehörigen Task auf dem mobilen Endgerät gespeichert

**Fehlerfälle:**

1. Der Benutzer beendet die Anwendung bevor die Aufgabe erfolgreich abgeschlossen wurde. Dies führt zum Abbruch der Messung.
2. Der Absturz der Anwendung führt ebenfalls zum Abbruch der Messung.

**Häufigkeit:**
Pro Task kann einmal Messgröße *Aufruf der Hilfefunktion* ermittelt werden.

**Erweiterung:**

1. Der Benutzer bricht die Aktion ab. Dies führt zum Abbruch der Messung der spezifischen Messgröße.

**Anforderungen:**

1. Effizienz:
   Die Speicherung der spezifischen Messgröße in die Datenbank oder die Protokollierung dürfen die Anwendung maximal um 250ms verzögern.

### A.2.5 Anzahl der Touch Events

**Titel:** **Anzahl der Touch Events**
**Akteur:** **Mobile Anwendung**
**Ziel:** **Anzahl der unterschiedlichen Touch Events messen**

**Auslöser:**
Der Benutzer berührt den Bildschirm und löst ein Touch Event.

**Vorbedingungen:**
Die Messgröße *Anzahl der Touch Events* ist vollständig in der Konfiguration für den Task definiert. Außerdem hat der Benutzer der Usability-Untersuchung zugestimmt.

**Nachbedingungen:**
Der Task wurde erfolgreich beendet.

**Erfolgsszenario:**

1. Benutzer beginnt mit dem Task
2. Benutzer beendet den Task erfolgreich
3. Der Wert der spezifischen Messgröße wird zusammen mit dem zugehörigen Task auf dem mobilen Endgerät gespeichert

**Fehlerfälle:**

1. Der Benutzer beendet die Anwendung bevor die Aufgabe erfolgreich abgeschlossen wurde. Dies führt zum Abbruch der Messung.
2. Der Absturz der Anwendung führt ebenfalls zum Abbruch der Messung.

**Häufigkeit:**
Pro Task können verschiedenen Touch Events ermittelt werden.

**Erweiterung:**

1. Der Benutzer bricht die Aktion ab. Dies führt zum Abbruch der Messung der spezifischen Messgröße.

**Anforderungen:**

1. Effizienz:
   Die Speicherung der spezifischen Messgröße oder die Protokollierung dürfen die Anwendung maximal um 250ms verzögern

### A.2.6 Anzahl der Key Events

**Titel:** **Anzahl der Key Events**
**Akteur:** **Mobile Anwendung**
**Ziel:** **Anzahl der unterschiedlichen Key Events messen**

**Auslöser:**
Der Benutzer löst mit den Tasten auf dem mobilen Endgerät ein Key Event aus.

**Vorbedingungen:**
Die Messgröße *Anzahl der Key Events* ist vollständig in der Konfiguration für den Task definiert. Außerdem hat der Benutzer der Usability-Untersuchung zugestimmt.

**Nachbedingungen:**
Der Task wurde erfolgreich beendet.

**Erfolgsszenario:**

1. Benutzer beginnt mit dem Task
2. Benutzer beendet den Task erfolgreich
3. Der Wert der spezifischen Messgröße wird zusammen mit dem zugehörigen Task auf dem mobilen Endgerät gespeichert

**Fehlerfälle:**

1. Der Benutzer beendet die Anwendung bevor die Aufgabe erfolgreich abgeschlossen wurde. Dies führt zum Abbruch der Messung.
2. Der Absturz der Anwendung führt ebenfalls zum Abbruch der Messung.

**Häufigkeit:**
Pro Task können verschiedene Key Events ermittelt werden

**Erweiterung:**

1. Der Benutzer bricht die Aktion ab. Dies führt zum Abbruch der Messung der spezifischen Messgröße.

**Anforderungen:**

1. Effizienz:
   Die Speicherung der spezifischen Messgröße oder die Protokollierung dürfen die Anwendung maximal um 250ms verzögern.

### A.2.7 Komplexität in der Navigation

**Titel:** **Komplexität in der Navigation**
**Akteur:** **Mobile Anwendung**
**Ziel:** **Komplexität in der Navigation der Anwendung ermitteln**

**Auslöser:**
Der Benutzer startet den Task.

**Vorbedingungen:**
Die Messgröße *Komplexität in der Navigation* ist vollständig in der Konfiguration für den Task definiert. Außerdem hat der Benutzer der Usability-Untersuchung zugestimmt.

**Nachbedingungen:**
Der Task wurde erfolgreich beendet.

**Erfolgsszenario:**

1. Benutzer beginnt mit dem Task
2. Im Task wird die Navigation durch die Menüs protokolliert
3. Benutzer beendet den Task erfolgreich
4. Der Wert der spezifischen Messgröße wird zusammen mit dem zugehörigen Task auf dem mobilen Endgerät gespeichert.

**Fehlerfälle:**

1. Der Benutzer beendet die Anwendung bevor die Aufgabe erfolgreich abgeschlossen wurde. Dies führt zum Abbruch der Messung.
2. Der Absturz der Anwendung führt ebenfalls zum Abbruch der Messung.

**Häufigkeit:**
Pro Task kann einmal die Messgröße *Komplexität in der Navigation* ermittelt werden.

**Erweiterung:**

1. Der Benutzer bricht die Aktion ab. Dies führt zum Abbruch der Messung der spezifischen Messgröße.

**Anforderungen:**

1. Effizienz:
   Die Speicherung der spezifischen Messgröße oder die Protokollierung dürfen die Anwendung maximal um 250ms verzögern.

## A.3 Protokollierung

**Titel:** **Protokollierung**
**Akteur:** **Mobile Anwendung**
**Ziel:** **Aktion des Benutzers protokollieren**

**Auslöser:**
Der Benutzer ruft eine Klasse/Methode auf, die eine Protokollierung auslöst.

**Vorbedingungen:**
Der Benutzer hat der Usability-Untersuchung zugestimmt.

**Nachbedingungen:**
Keine

**Erfolgsszenario:**

1. Der Benutzer startet die Anwendung
2. Protokolldatei wird geöffnet/ erstellt
3. Protokollierung der Schritte bei entsprechendem Aufruf
4. Protokolldatei wird beim Beenden der Anwendung geschlossen

**Anforderungen:**

1. Effizienz:
   Das Schreiben in die Protokolldatei sorgt für eine größere Verzögerungszeit der Anwendung. Die Verzögerungszeit sollte bei maximal 250ms liegen.
2. Zuverlässigkeit (Robustheit):
   Bei Absturz oder Fehlerfall soll die Protokollierung für eine spätere Fehlersuche/ Auswertung bestehen bleiben.
3. Sicherheit (Security):
   Die Protokolldatei muss vor Zugriff von Unbefugten geschützt werden. Die Anwendung und die Usability-Komponente sollten nur Zugriff auf die Datei haben.

## A.4 Benutzerinteraktion

**Titel:** **Benutzerinteraktion**
**Akteur:** **Mobile Anwendung**
**Ziel:** **Zustimmung vom Benutzer für die Usability-Untersuchung der Anwendung**

**Auslöser:**
Beim Starten und Beenden der Anwendung.

**Vorbedingungen:**
Keine

**Nachbedingungen:**
Keine

**Erfolgsszenario:**

1. Der Benutzer hat der Usability-Untersuchung zugestimmt
2. Der Benutzer möchte die Anwendung schließen
3. Fragebogen zur Bewertung der Anwendung wird vom Benutzer ausgefüllt
4. Anwendung wird geschlossen

**Alternative:**

1. Der Benutzer stimmt der Usability-Untersuchung nicht zu
2. Die Usability-Komponente bleibt inaktiv
3. Anwendung wird geschlossen

## A.5 Auswertung

### A.5.1 Auswertung der Daten einsehen

**Titel:** **Auswertung der Daten einsehen**
**Akteur:** **Entwickler/Usability-Experte**
**Ziel:** **Der Entwickler kann die aufbereiteten Daten und Protokolldateien einsehen**

**Auslöser:**
Der Entwickler ruft die Webseite mit den aufbereiteten Daten und Protokolldateien auf.

**Vorbedingungen:**
Daten wurden vom Benutzer an den Server gesendet.

**Nachbedingungen:**
Keine

**Erfolgsszenario:**

1. Webseite auf dem Server wird vom Entwickler aufgerufen
2. Nach erfolgreicher Authentifizierung erhält der Entwickler Zugriff auf die Daten

**Anforderungen:**

1. Effizienz:
   Der Aufruf der Webseite darf beim Entwickler maximal 2 Sekunden dauern.
2. Sicherheit (Security):
   Die Daten auf dem Webserver müssen vor unbefugten Zugriff von außen geschützt werden. Der Entwickler muss sich auf dem Server authentifizieren, um Zugriff auf die Daten zu bekommen.

### A.5.2 Auswertung der Daten

**Titel:** **Auswertung der Daten**
**Akteur:** **Entwickler**
**Ziel:** **Die gesammelten Daten über Messgrößen werden ausgewertet**

**Auslöser:**
Nachdem erfolgreichem Senden der Daten an den Server

**Vorbedingungen:**
Erfolgreiches Senden der Daten

**Nachbedingungen:**
Keine

**Erfolgsszenario:**

1. Daten werden für den Entwickler aufbereitet

**Fehlerfälle:**

1. Fehlerhafte oder nicht vollständige Daten werden für die Auswertung nicht berücksichtigt

**Häufigkeit:**
Nach jedem erfolgreichem Senden der Daten wird eine Auswertung der Daten vorgenommen.

**Anforderungen:**

1. Zuverlässigkeit (Korrektheit):
   Fehler bei der Auswertung der Daten dürfen nicht in die Statistik der ausgewerteten Daten eingehen.
2. Sicherheit (Safety):
   Die Daten über die Messgrößen der jeweiligen Tasks und die Protokolldatei dürfen bei der Auswertung nicht verändert werden. Ungültige Daten werden für die Auswertung nicht berücksichtigt.
3. Sicherheit (Security):
   Die ausgewerteten Daten werden in einem für den Entwickler eingerichteten Bereich abgelegt. Der Entwickler kann über den Webbrowser mittels Authentifizierung diese Daten einsehen.

## A.6 Kommunikation

### A.6.1 Senden der Daten

**Titel:** **Senden der Daten**
**Akteur:** **Mobile Anwendung**
**Ziel:** **Die Daten der Messgrößen und die Protokolldatei werden an den Server gesendet**

**Auslöser:**
Der Benutzer bestätigt das Senden der Daten.

**Vorbedingungen:**

- Der Benutzer hat die Teilnahme an der Usability-Untersuchung der Anwendung bestätigt.
- Eine entsprechende Bandbreite z.B. UMTS, WLAN zum Senden der Daten ist vorhanden.
- Der Server und die Schnittstelle sind vom Internet aus erreichbar.

**Nachbedingungen:**
Keine

**Erfolgsszenario:**

1. Der Benutzer erhält in unregelmäßigen Abständen nach Beendigung der Tasks in der Anwendung eine Anfrage zum Senden der Daten
2. Der Benutzer bestätigt das Senden der Daten
3. Use Case „Empfang der Daten“
4. Daten senden erfolgreich

**Alternative(a) ab 2.:**

1. Der Benutzer lehnt das Senden der Daten ab
2. Benutzer erhält Hinweis auf manuelles Senden der Daten

**Alternative(b):**

- In der Anwendung kann der Benutzer das Senden der Daten manuell ausführen.

**Fehlerfälle:**

1. Das Senden der Daten schlägt fehl. Ein weiter Versuch wird gestartet. Bei erneutem Fehlschlag wird der Benutzer auf das manuelle Senden hingewiesen.
2. Bandbreite nicht vorhanden. Erneuter Hinweis auf manuelles Senden.

**Anforderungen:**

1. Effizienz:
   Die Daten über die Messgrößen der jeweiligen Tasks und die Protokolldateien dürfen maximal 500KB groß sein. Die Zeit für die Übertragung darf bei maximal 20sec liegen.
2. Benutzbarkeit (Ergonomie):
   Dem Benutzer sollte beim Senden der Daten ein Statusbalken angezeigt werden. Der Benutzer bekommt eine Rückmeldung über die Dauer des Vorgangs.
3. Sicherheit (Safety):
   Das Senden der Daten darf die enthaltenden Daten nicht verändern.
4. Sicherheit (Security):
   Das Senden der Daten muss gegen Unbefugte abgesichert werden. Die Kommunikation erfolgt hierbei über ein gesicherten Kanal zum Beispiel HTTPS [wikipedia (2009), Force (2000)].

### A.6.2 Empfang der Daten

**Titel:** **Empfang der Daten**
**Akteur:** **Mobile Anwendung**
**Ziel:** **Die Daten der Messgrößen und die Protokolldatei vom mobilen Endgerät werden vom Server empfangen**

**Auslöser:**
Die Daten werden an den Server gesendet.

**Vorbedingungen:**
Der Server und die Schnittstelle sind vom Internet aus erreichbar.

**Nachbedingungen:**
Keine

**Erfolgsszenario:**

1. Daten werden an den Server gesendet
2. Daten senden erfolgreich

3. Use Case „Auswertung der Daten“

**Fehlerfälle:**
Das Senden der Daten schlägt fehl. Das mobile Endgerät wird vom Server über die fehlerhafte Übertragung informiert.

**Anforderungen:**

1. Sicherheit (Security):
   Das Empfangen der Daten muss gegen Unbefugte abgesichert werden. Die Kommunikation erfolgt hierbei über ein gesicherten Kanal zum Beispiel HTTPS [wikipedia (2009), Force (2000)].

# B Sourcecode

## B.1 Speicherung der Daten im XML-Format

Das erste Beispiel zeigt das XML-Format für die Messergebnisse.

```
<?xml version="1.0" encoding="UTF−8"?>
<task name="test">
  <measurement name="TaskTime">
    <data name="TIME" value="16344"/>
  </measurement>
  <measurement name="KeyEvent">
    <data name="DPAD_CENTER" value="1"/>
    <data name="DPAD_DOWN" value="6"/>
    <data name="DPAD_LEFT" value="1"/>
    <data name="DPAD_RIGHT" value="1"/>
    <data name="DPAD_UP" value="5"/>
  </measurement>
  <measurement name="TouchEvent">
    <data name="ON_DOWN" value="15"/>
    <data name="ON_FLING" value="4"/>
    <data name="ON_LONGPRESS" value="0"/>
    <data name="ON_SCROLL" value="5"/>
    <data name="ON_SHOWPRESS" value="1"/>
    <data name="ON_SINGLETAPUP" value="10"/>
  </measurement>
</task>
```

Das zweite Beispiel zeigt das XML-Format für die Feedback-Daten an.

```
<?xml version="1.0" encoding="UTF−8"?>
<feedbacks>
  <feedback answer="Stimme voll zu" question="Ist die Anwendung gelungen?"/>
  <feedback answer="Stimme zu" question="Es gab keinerlei Probleme bei der Bedienung"/>
</feedbacks>
```

## B.2 Konfiguration im XML-Format

```
<?xml version="1.0" encoding="UTF−8"?>
<config serverURL="IP" serverPort="Port">
<tasks>
  <task name="taskName">
    <measurement name="MeasurementName"/>
    <measurement name="MeasurementName"/>
    <measurement name="MeasurementName"/>
     ...
  </task>
   ...
</tasks>
</config>
```

## B.3 GUI Layout des Fragebogens

```
public LinearLayout initFeedbackView(CharSequence text) {
   LayoutParams lp = new LayoutParams(LayoutParams.
      FILL_PARENT,
      LayoutParams.FILL_PARENT);
   LayoutParams wrap = new LayoutParams(LayoutParams.
      WRAP_CONTENT,
      LayoutParams.WRAP_CONTENT);

   LinearLayout ol = new LinearLayout(this);
   ol.setLayoutParams(lp);

   LinearLayout il = new LinearLayout(this);
   il.setLayoutParams(wrap);
   il.setOrientation(LinearLayout.VERTICAL);

   tv = new TextView(this);
   tv.setLayoutParams(wrap);
   tv.setText(text);

   RadioGroup rg = new RadioGroup(this);
   rg.setLayoutParams(wrap);
```

```
    rg.setOrientation(LinearLayout.VERTICAL);

    for (int i = 0; i < a.length; i++) {
      RadioButton rb = new RadioButton(this);
      rb.setText(a[i]);
      rb.setLayoutParams(wrap);
      rg.addView(rb);
    }

    il.addView(tv);
    il.addView(rg);
    ol.addView(il);
    checkBundle(text, rg);
    rg.setOnCheckedChangeListener(this);
    return ol;
  }
```

## B.4 Konkrete Konfiguration der Usability-Komponenten in Learn!

```
<?xml version="1.0" encoding="UTF−8"?>
<config serverURL="192.168.0.42" serverPort="9999">
<tasks>
  <task name="allgemein">
    <measurement name="FailureRate"/>
    <measurement name="ExceptionRate"/>
    <measurement name="HelpFunctionRate"/>
  </task>
  <task name="task1">
    <measurement name="TaskTime"/>
    <measurement name="KeyEvent"/>
    <measurement name="TouchEvent"/>
  </task>
  <task name="task2">
    <measurement name="TaskTime"/>
    <measurement name="KeyEvent"/>
    <measurement name="TouchEvent"/>
    <measurement name="Complexity"/>
```

```
  </task>
</tasks>
</config></config>
```

# Tabellenverzeichnis

# Abbildungsverzeichnis

# Abkürzungsverzeichnis

**AAPT** Android Asset Packaging Tool

**ADB** Android Debug Bridge

**API** Application Programming Interface

**DDMS** Dalvik Debug Monitoring Service

**GPRS** General Packet Radio Service

**HTTPS** Hypertext Transfer Protocol Secure

**ISO** International Organisation for Standardisation

**JVM** Application Programming Interface

**SDK** Software Development Kit

**UI** User Interface

**UMTS** Universal Mobile Telecommunications System

**WLAN** Wireless Local Area Network

**XML** Extensible Markup Language

# Literaturverzeichnis


[Alliance a] ALLIANCE, Open H.: *Android*. – URL http://developer.android.com/

[Alliance b] ALLIANCE, Open H.: *Open Handset Alliance*. – URL http://www.openhandsetalliance.com/

[Alliance 2010] ALLIANCE, Open H.: *User Interface Guidelines*. 2010. – URL http://developer.android.com/guide/practices/ui_guidelines/index.html

[amazon.de ] AMAZON.DE: *amazon.de*. – URL www.mazon.de

[Balagtas-Fernandez und Hussmann 2009] BALAGTAS-FERNANDEZ, Florence ; HUSSMANN, Heinrich: A Methodology and Framework to Simplify Usability Analysis of Mobile Applications. In: *ASE '09: Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*. Washington, DC, USA : IEEE Computer Society, 2009, S. 520–524. – ISBN 978-0-7695-3891-4

[Bevan und Curson 1997] BEVAN, Nigel ; CURSON, Ian: *Methods for Measuring Usability*. 1997. – URL http://www.nigelbevan.com/papers/meatut97.pdf

[Bevan und Macleod 1994] BEVAN, Nigel ; MACLEOD, Miles: Usability measurement in context. In: *Behaviour and Information Technology* (1994), Nr. 13, S. 132–145

[Bundesamt 2009] BUNDESAMT, Statistisches: *Zuhause in Deutschland*. 2009. – URL https://www-ec.destatis.de/csp/shop/sfg/bpm.html.cms.cBroker.cls?cmspath=struktur,vollanzeige.csp&ID=1023672

[dict.cc ] DICT.CC: *dict.cc*. – URL http://www.dict.cc/?s=usability

[Duh u. a. 2006] DUH, Henry Been-Lirn ; TAN, Gerald C. B. ; CHEN, Vivian Hsueh-hua: Usability evaluation for mobile device: a comparison of laboratory and field tests. In: *MobileHCI '06: Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*. New York, NY, USA : ACM, 2006, S. 181–186. – ISBN 1-59593-390-5

[Eclipse ] ECLIPSE: *Eclipse*. – URL http://www.eclipse.org

[Eichinger ] EICHINGER, Armin: *Usability*. – URL http://www.uni-regensburg.de/Fakultaeten/phil_Fak_II/Psychologie/Doktoranden/absolventen/eichinger_armin/u-vorbemerkungen.html

[Erdinger ] ERDINGER, Heike: *Was ist Usability?*. – URL http://www.javajim.de/theorietank/usability/wasistusability.html

[Esser 2001] ESSER, Friedrich: *Java 2*. Galileo Computing, 2001. – ISBN 3-9343-5866-7

[Force 2000] FORCE, The Internet Engineering T.: *HTTP Over TLS*. 2000. – URL http://tools.ietf.org/html/rfc2818

[Gartner 2010] GARTNER: *Gartner Says Worldwide Mobile Phone Sales to End Users Grew 8 Per Cent in Fourth Quarter 2009; Market Remained Flat in 2009*. 2010. – URL http://www.gartner.com/it/page.jsp?id=1306513

[Google 2009] GOOGLE: *Google Collections Library 1.0*. 2009. – URL http://code.google.com/p/google-collections/

[Google 2010] GOOGLE: *Android 1.6*. 2010. – URL http://developer.android.com/sdk/index.html

[Gunawardana u. a. 2010] GUNAWARDANA, Asela ; PAEK, Tim ; MEEK, Christopher: Usability guided key-target resizing for soft keyboards. In: *IUI '10: Proceeding of the 14th international conference on Intelligent user interfaces*. New York, NY, USA : ACM, 2010, S. 111–118. – ISBN 978-1-60558-515-4

[Ham u. a. 2006] HAM, Dong-Han ; HEO, Jeongyun ; FOSSICK, Peter ; WONG, William ; PARK, Sanghyun ; SONG, Chiwon ; BRADLEY, Mike: Conceptual framework and models for identifying and organizing usability impact factors of mobile phones. In: *OZCHI '06: Proceedings of the 18th Australia conference on Computer-Human Interaction*. New York, NY, USA : ACM, 2006, S. 261–268. – ISBN 1-59593-545-2

[Helfenbein und Claußner 2006] HELFENBEIN, Sascha ; CLAUSSNER, Daniel: *Usability Testing*. 2006. – URL http://www.tu-chemnitz.de/phil/psych/professuren/allpsy1/lehre/Ergonomie/Usability%20Testing.pdf

[Herczeg 1994] HERCZEG, Michael: *Software-Ergonomie. Grundlagen der Mensch-Computer-Kommunikation*. Oldenbourg Verlag, 1994. – ISBN 3-89319615-3

[htc.com 2009] HTC.COM: *HTC*. 2009. – URL http://www.htc.com/

[Hussain und Ferneley 2008] HUSSAIN, Azham ; FERNELEY, Elaine: Usability metric for mobile application: a goal question metric (GQM) approach. In: *iiWAS '08: Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*. New York, NY, USA : ACM, 2008, S. 567–570. – ISBN 978-1-60558-349-5

[Inc. 2010] INC., Apple: *IPhone Human Interface Guidelines*. 2010. – URL http://developer.apple.com/iphone/library/documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html

[InfoWissWiki ] INFOWISSWIKI: *Usability Testing und Engineering*. – URL http://wiki.infowiss.net/Usability_Testing_und_Engineering

[Irion 2002] IRION, Thomas: Collection, Presentation and Analysis of Multimedia Data with Computers. In: *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research* 3 (2002), Nr. 2. – URL http://www.qualitative-research.net/index.php/fqs/article/view/855/1859. – ISSN 1438-5627

[IT-Times 2009] IT-TIMES: *Research In Motion sieht im App Store Geschäft seine Zukunft*. 2009. – URL http://www.it-times.de/news/nachricht/seite/1/datum/2009/11/23/research-in-motion-sieht-im-app-store-geschaeft-seine-zukunft/

[Ivory und Hearst 2001] IVORY, Melody Y. ; HEARST, Marti A.: The state of the art in automating usability evaluation of user interfaces. In: *ACM Comput. Surv.* 33 (2001), Nr. 4, S. 470–516. – ISSN 0360-0300

[Jokela u. a. 2006] JOKELA, Timo ; KOIVUMAA, Jussi ; PIRKOLA, Jani ; SALMINEN, Petri ; KANTOLA, Niina: Methods for quantitative usability requirements: a case study on the development of the user interface of a mobile phone. In: *Personal Ubiquitous Comput.* 10 (2006), Nr. 6, S. 345–355. – ISSN 1617-4909

[Kahlbrandt 2001] KAHLBRANDT, Bernd: *Software Engineering mit der Unified Modeling Language*. Springer, 2001. – ISBN 3-5404-1600-5

[Karlson u. a. 2010] KARLSON, Amy K. ; IQBAL, Shamsi T. ; MEYERS, Brian ; RAMOS, Gonzalo ; LEE, Kathy ; TANG, John C.: Mobile taskflow in context: a screenshot study of smartphone usage. In: *CHI '10: Proceedings of the 28th international conference on Human factors in computing systems*. New York, NY, USA : ACM, 2010, S. 2009–2018. – ISBN 978-1-60558-929-9

[Keijzers u. a. 2008] KEIJZERS, Jeroen ; OUDEN, Elke den ; LU, Yuan: Usability benchmark study of commercially available smart phones: cell phone type platform, PDA type platform and PC type platform. In: *MobileHCI '08: Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*. New York, NY, USA : ACM, 2008, S. 265–272. – ISBN 978-1-59593-952-4

[Keith 2009] KEITH, Andrew: *Human-ComputerInteraction*. 2009. – URL http://courses.iicm.tugraz.at/hci/hci.pdf

[Kjeldskov und Graham 2003] KJELDSKOV, Jesper ; GRAHAM, Connor: *A Review of Mobile HCI Research Methods*. 2003. – URL http://www.cs.aau.dk/~jesper/papers/MobileHCI03-final.pdf

[Kjeldskov u. a. 2004a] KJELDSKOV, Jesper ; SKOV, Mikael B. ; ALS, Benedikte S. ; HØEGH, Rune T.: *Is it Worth the Hassle? Exploring the Added Value of Evaluating the Usability of Context-Aware Mobile Systems in the Field*. 2004. – URL http://www.cs.aau.dk/~jesper/papers/MobileHCI04-final.pdf

[Kjeldskov u. a. 2004b] KJELDSKOV, Jesper ; SKOV, Mikael B. ; ALS, Benedikte S. ; HØEGH, Rune T.: Is It Worth the Hassle? Exploring the Added Value of Evaluating the Usability of Context-Aware Mobile Systems in the Field. In: *Mobile HCI*, 2004, S. 61–73

[Koskinen u. a. 2008] KOSKINEN, Emilia ; KAARESOJA, Topi ; LAITINEN, Pauli: Feel-good touch: finding the most pleasant tactile feedback for a mobile touch screen button. In: *IMCI '08: Proceedings of the 10th international conference on Multimodal interfaces*. New York, NY, USA : ACM, 2008, S. 297–304. – ISBN 978-1-60558-198-9

[Krug 2000] KRUG, Steve: *Don´t Make Me Think*. New Riders, 2000. – ISBN 0-7897-2310-7

[Macleod und Rengger 1993] MACLEOD, Miles ; RENGGER, Ralph: *The Development of DRUM:A Software Tool for Video-assisted Usability Evaluation*. 1993. – URL http://www.nigelbevan.com/papers/drum93.pdf

[Mobilfunk 2009] MOBILFUNK, Informationszentrum: *Mobilfunkteilnehmer in Deutschland*. 2009. – URL http://www.izmf.de/html/de/46272.html

[Mosemann und Kose 2009] MOSEMANN, Heiko ; KOSE, Matthias: *Android - Anwendungen für das Handy-Betriebssystem erfolgreich programmieren*. Hanser Fachbuch, 2009. – ISBN 978-3-446-41728-1

[Nielsen 1994] NIELSEN, Jakob: *Usability Engineering*. Morgan Kaufmann, 1994. – ISBN 0-12-518406-9

[Nielsen 2009] NIELSEN, Jakob: *Mobile Usability*. 2009. – URL http://www.useit.com/alertbox/mobile-usability.html

[Nokia 2010] NOKIA: *Design and User Experience Library v1.9*. 2010. – URL http://library.forum.nokia.com/index.jsp?topic=/Design_and_User_Experience_Library/GUID-7722CB80-BCFC-487F-8D6C-E3C5AD0ADC1F.html

[Oestereich 2006] OESTEREICH, Bernd: *Analyse und Design mit UML 2.1 - Objektorientierte Softwareentwicklung* -. 8. aktualisierte Auflage. Oldenbourg, 2006. – ISBN 978-3-446-41728-1

[Park u. a. 2008] PARK, Yong S. ; HAN, Sung H. ; PARK, Jaehyun ; CHO, Youngseok: Touch key design for target selection on a mobile phone. In: *MobileHCI '08: Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*. New York, NY, USA : ACM, 2008, S. 423–426. – ISBN 978-1-59593-952-4

[POI 2009] POI, Apache: *Apache POI - the Java API for Microsoft Documents*. 2009. – URL http://poi.apache.org/

[Priebs 2006] PRIEBS, Markus: *Evaluation und Usability von mobilen Benutzerschnittstellen*. 2006. – URL http://www-mmt.inf.tu-dresden.de/Lehre/Wintersemester_05_06/Proseminar/Proceedings/EvaluationundUsabilityvonmobilenBenutzerschnittstellen.pdf

[Ryan und Gonsalves 2005] RYAN, Caspar ; GONSALVES, Atish: The effect of context and application type on mobile usability: an empirical study. In: *ACSC '05: Proceedings of the Twenty-eighth Australasian conference on Computer Science*. Darlinghurst, Australia, Australia : Australian Computer Society, Inc., 2005, S. 115–124. – ISBN 1-920-68220-1

[Saarland ] SAARLAND, Uni: *Arbeitsbereich Usability Engineering*. – URL http://usability.is.uni-sb.de/

[Schuhmacher ] SCHUHMACHER, Dr. J.: *Usability*. – URL http://controlling21.de/ergonomie/theorie/grundlagen/usability.htm

[Sourceforge 2010] SOURCEFORGE: *FIND AND DEVELOP OPEN SOURCE SOFTWARE*. 2010. – URL http://sourceforge.net/

[sparhandy.de 2009] SPARHANDY.DE: *Sparhandy*. 2009. – URL http://www.sparhandy.de/

[Sun Microsystems ] SUN MICROSYSTEMS, Inc: *Java*. – URL http://developers.sun.com/

[Vaitukaitis 2010] VAITUKAITIS, Vytautas: *Learn!* 2010. – URL http://www.cl.cam.ac.uk/research/dtg/language/

[Voigt 2009] VOIGT, Bastian: *Software entwickeln mit dem AndroidSDK*. 2009. – URL www.jughh.org/download/attachments/3637315/Android.pdf

[Wache ] WACHE, Michael: *Grundlagen von E-Learning*. – URL http://www1.bpb.de/methodik/87S2YN,0,0,Grundlagen_von_eLearning.html

[Wiki 2010] WIKI: *Mobile Design*. 2010. – URL http://wiki.forum.nokia.com/index.php/Category:Mobile_Design

[Wikipedia a] WIKIPEDIA: *Bibtex*. – URL http://de.wikipedia.org/wiki/BibTeX

[Wikipedia b] WIKIPEDIA: *Usability-Labor*. – URL http://de.wikipedia.org/wiki/Usability-Labor

[wikipedia 2009] WIKIPEDIA: *Hypertext Transfer Protocol Secure*. 2009. – URL http://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol_Secure

[Wikipedia 2010] WIKIPEDIA: *Android (operating system)*. 2010. – URL http://en.wikipedia.org/wiki/Android_(operating_system)

[Wixler und Schwertel 2007] WIXLER, Robert ; SCHWERTEL, Dennis: *Ausarbeitung:Android*. 2007. – URL www.mi.fh-wiesbaden.de/~barth/mobile/ws0708/Android.pdf

[XStream 2008] XSTREAM: *XStream*. 2008. – URL http://xstream.codehaus.org/index.html

[ZDNet 2009] ZDNET: *Marktforscher erwarten starkes Wachstum bei App-Downloads*. 2009. – URL http://www.zdnet.de/news/mobile_wirtschaft_marktforscher_erwarten_starkes_wachstum_bei_app_downloads_story-39002365-41516008-1.htm

# Versicherung über Selbstständigkeit

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

| | |
|---|---|
| Hamburg, 13. Juli 2010 | |
| Ort, Datum | Unterschrift |