Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Seminararbeit Anwendungen 1

Thomas Preisler

Vorgehensmodelle für mobile Anwendungen

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Thomas Preisler

Vorgehensmodelle für mobile Anwendungen

Seminararbeit eingereicht im Rahmen der Veranstaltung Anwendungen 1
im Studiengang Master of Science Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuer : Prof. Dr. Olaf Zukunft

Abgegeben am 20. Juli 2008

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Die Entwicklung von mobilen Anwendungen hat in den letzten Jahren zunehmend an Bedeutung gewonnen. Die Anzahl und die Leistungsmöglichkeit von mobilen Geräten, wie Mobiltelefonen und PDAs, ist stetig gestiegen. Für die Entwicklung von mobilen Anwendungen existieren unterschiedliche Programmiersprachen und Plattformen, aber die wichtigen Bereiche Testen und Usability–Untersuchungen wurden bisher weitgehend vernachlässigt (vgl. Delamaro u. a. (2006)).

Im Gegensatz dazu existieren für die Entwicklung von immobilen[1] Anwendungen ausgereifte Vorgehensmodelle. Die Bereiche Testen und Usability–Untersuchungen sind in Forschung, Lehre und Praxis weitreichend erforscht und ebenso verbreitet. Als Beispiele in Forschung und Lehre können das „Usability–Lab“ und das Wahlpflichtfach „Certified Tester“ an der HAW Hamburg genannt werden.

Diese Vorgehensmodelle lassen sich aber nur bedingt auf mobile Anwendungen übertragen. Durch die Mobilität der Geräte entstehen neue Herausforderungen im Bereich Testen und Usability, die die vorhandenen Vorgehensmodelle nicht alle abdecken. Neue Vorgehensmodelle bzw. Abwandlungen der Vorhandenen werden erforderlich. Einige davon werden im Rahmen dieser Arbeit vorgestellt.

1.1 Abgrenzung mobile Anwendungen

Vorgehensmodelle für mobile Anwendungen bezieht sich im Rahmen dieser Arbeit auf Anwendungen, die für mobile Geräte, wie die eingangs erwähnten Mobiltelefone und PDAs entwickelt werden. Der Bereich der Anwendungen für mobile, eingebettete Systeme wird in dieser Arbeit nicht betrachtet. Eine solche Betrachtung würde den inhaltlichen Rahmen dieser Arbeit überschreiten und hat keinerlei Relevanz für das Projektthema (Pervasive Gaming) im nächsten Semester.

[1] Im folgenden werden Anwendungen, die nicht für mobile Geräte, wie Mobiltelefone und PDAs entwickelt werden, als immobile Anwendungen bezeichnet.

1.2 Gliederung der Arbeit

Im zweiten Kapitel werden zunächst die Grundlagen des Testens von mobilen Anwendungen erklärt, anschließend werden mit *MobileTest* und *JaBUTi/ME* zwei konkrete Lösungen vorgestellt. Das dritte Kapitel beschäftigt sich mit Usability–Untersuchungen für mobile Anwendungen. Dabei wird auf die generellen Herausforderungen einer solchen Usability-Untersuchung und die Probleme von Feld- und Labortests eingegangen.

2 Testen von mobilen Anwendungen

2.1 Grundlagen

Mobile Anwendungen (Mahmoud, 2002) müssen, wie alle anderen Anwendungen auch, getestet werden, um die korrekte Funktionsweise sicherzustellen. Im Bereich der mobilen Anwendungen ist Testen von besonderer Bedeutung, da die Arbeitsbedingungen und die Umgebungen stärker variieren, als bei den meisten anderen Anwendungen. Die Entwicklung von mobilen Anwendungen findet meist auf leistungsstarken Computern statt. Zur Entwicklungszeit werden die mobilen Anwendungen meist in einem Emulator getestet und erst dann findet das Deployment auf die mobilen Geräte statt. Die mobilen Geräte selbst weisen dabei häufig sehr unterschiedliche technische Charakteristika auf. Natürlich reicht es nicht aus die mobilen Anwendungen nur in Emulatoren zu testen, da diese keine komplette Kompatibilität mit dem Zielgerät garantieren können. Auch bildet ein Emulator i. d. R. nicht alle Umgebungen ab, in denen die Anwendung zum Einsatz kommen kann.

Die Anforderungen (Bo u. a., 2007), welche Benutzer an die Stabilität und Zuverlässigkeit von mobilen Anwendungen stellen sind oftmals höher, als bei immobilen Anwendungen. Die Benutzer erwarten bei ihren mobilen Geräten deutlich geringere Auswahlraten, als bei ihren Heimcomputern. Mobile Anwendungen sind aber von der Funktionalität nicht weniger komplex, als immobile Anwendungen (vgl. Binder und Hanlon (2005)).

2.1.1 Problemfelder

Beim Testen von mobilen Anwendungen treten eine Reihe von Problemen auf (Bo u. a., 2007), die bei immobilen Anwendungen keine Rolle spielen. Diese Probleme resultieren aus den technischen Einschränkungen der mobilen Geräte sowie der mobilen Nutzung.

Beschränkte Hardware–Ressourcen: Aufgrund der beschränkten Hardware–Ressourcen der mobilen Geräte ist es schwierig *auf* diesen zu testen. So ist z. B. die Eingabe einer großen Menge von Testdaten, über die meist beschränkten Eingabemöglichkeiten, unkomfortabel. Ein vernünftiger Test benötigt aber viele Testdaten. Außerdem weisen die

mobilen Geräte meist nur beschränkte Möglichkeiten der Ausgabe und Protokollierung auf, was die Auswertung der Testergebnisse erschwert.

Umgebung der Zielgeräte: Die Umgebungen der Zielgeräte (wireless signals, context-sensitive-information) sind komplex. Ferner hängen die Konditionen der Umgebung teilweise von der Anzahl und den Tätigkeiten der Benutzer ab. (Jedes Jahr an Silvester werden kurz nach Mitternacht Unmengen von Neujahrsgrüßen per SMS verschickt, was teilweise zu starken Beeinträchtigungen der Mobilfunknetze führt.) Beim Testen müssen möglichst alle Veränderungen der Umgebung berücksichtigt werden, um sicherzustellen dass die Anwendung in jeder Umgebung korrekt funktioniert.

Vielfalt der Geräte: Die Vielfalt der mobilen Geräte erschwert u. U. die Wiederverwendbarkeit und Wartbarkeit der Testfälle.

Interaktives Verhalten: Das Verhalten von mobilen Geräten ist hochgradig interaktiv. Die Geräte akzeptieren ständig Eingaben vom Benutzer, verarbeiten diese und erwarten vom Benutzer weitere Eingaben. Diese Interaktivität macht es schwierig die Aktionen eines Benutzers vorherzusehen, was eine automatische Generierung von Testszenarien, welche alle realen Benutzungsszenarien abdecken, erschwert.

2.1.2 Testkonzepte

Bevor im folgenden mit *MobileTest* und *JaBUTi/ME* zwei konkrete Ansätze zum Testen von mobilen Anwendungen vorgestellt werden, werden die Testkonzepte *Black Box Test* und *White Box Test* erläutert.

Black Box Test

Ein funktionaler Test (Mahmoud, 2002), der das von außen sichtbare Verhalten prüft und dabei keine Kenntnis über die Programmlogik hat. Bei einem Black Box Test wird der Quellcode der Anwendung ignoriert und es wird nur überprüft, ob eine Eingabe zu der gewünschten Ausgabe/Aktion führt.

Abbildung 2.1: Black Box Testing

White Box Test

Ein strukturbezogener Test (Mahmoud, 2002), der den Kontrollfluss prüft. Kenntnisse über die Programmlogik und den Quellcode sind vorhanden. Das Ziel eines White Box Tests ist es, Fehler in der Implementierung zu finden, dafür werden alle Pfade durch den Quellcode mindestens einmal durchlaufen.

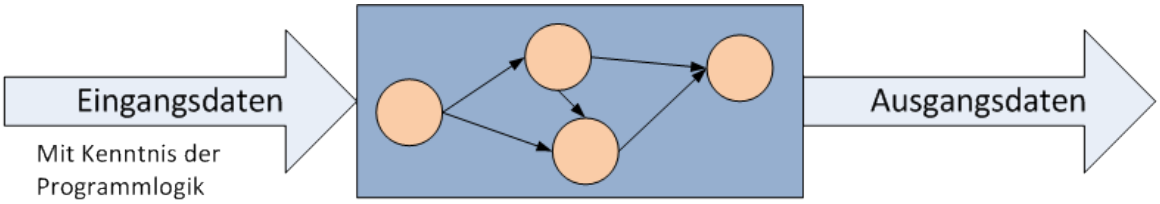

Abbildung 2.2: White Box Test

2.2 Black Box Test: MobileTest

MobileTest (Bo u. a., 2007) ist ein Programm für automatisierte Black Box Tests auf mobilen Geräten. Mit MobileTest ist es möglich wartbare und wiederverwendbare Testfälle für Anwendungen auf verschiedenen, mobilen Geräten zu erzeugen. Dabei werden die Testskripte, die den Ablauf eines Testfalls vorgeben, von einem Tester an einem Standard–Computer mit MobileTest entwickelt. Auf den mobilen Zielgeräten laufen mobile Agenten, die diese Testfälle dann ausführen. Die Ergebnisse werden von den mobilen Agenten wieder an den Standard–Computer übertragen und dort analysiert.

Die zwei wichtigsten Zielsetzungen von MobileTest sind:

- Die Benutzung von Bildvergleich und Texterkennung zur Ermittlung des Zustandes einer Anwendung soll minimiert werden. Diese Mechanismen werden von vielen gängigen Testprogrammen verwendet, eignen sich aber nicht um wiederverwendbare Testfälle zu erstellen. Unterschiedliche mobile Geräte verwenden unterschiedliche Darstellungen, sodass eine einheitliche Zustandserkennung über die Darstellung auf dem Bildschirm nicht möglich ist.
- MobileTest bietet Mechanismen, um die Umgebung des mobilen Gerätes zu kontrollieren, so ist es möglich das Gerät bzw. die Anwendung in verschiedenen Umgebungen zu testen, ohne das Gerät physikalisch bewegen zu müssen.

Um die Benutzung von Bildvergleich und Texterkennung zu minimieren, benutzt MobileTest die sogenannten „sensitive events“. Dieses sind Ereignisse, die auf dem mobilen Zielgerät bei einem bestimmten Testfall auftreten können. Mögliche Beispiele für einen solchen „sensitive event“ sind zum Beispiel: „eingehender Anruf“, „Telefonbuchspeicher voll“, „SMS empfangen“ usw..

2.2.1 Architektur

MobileTest besteht aus vier Schichten (vgl. Abb. 2.3):

1. User Interface Layer: Diese Schicht läuft auf dem Standard–Computer und interagiert mit dem Benutzer/Tester. Innerhalb dieser Schicht befindet sich der Test Script Editor, mit dem die Testfälle entwickelt werden. Außerdem hat der Tester hier die Möglichkeit die Testfälle zu verwalten und zu konfigurieren.
2. Test Control Layer: Diese Schicht läuft ebenfalls noch auf dem Standard–Computer. Hier werden die erzeugten Testskripte ausgeführt und entsprechende, simulierte Eingaben für das Zielgerät erzeugt. Neben den Modulen zur Überwachung des Tests und zur Verifikation der Resultate, befindet sich in dieser Schicht auch das Modul um die Umgebung des mobilen Gerätes zu kontrollieren.
3. Communication Layer: Diese Schicht verbindet die Test Control Layer mit der Device Agent Layer und damit den Standard–Computer mit dem mobilen Gerät. Daher läuft diese Schicht auf beiden Geräten. Verschiedene Netzwerkprotokolle ermöglichen eine Kommunikation mit unterschiedlichen mobilen Geräten.
4. Device Agent Layer: Der Mobile Agent, der auf dem mobilen Zielgeräten läuft. Der Agent empfängt die simulierten Eingaben, führt diese aus und sendet die Resultate zurück. Diese Schicht macht es möglich, dass die erzeugten Testfälle wiederverwendbar sind und auf unterschiedlichen, mobilen Zielgeräten verwendet werden können. Der mobile Agent bildet eine Abstraktionsschicht über den gerätespezifischen Eigenschaften der Hardware. Der Agent bildet dabei die simulierten Eingaben, die er über die Communication Layer empfängt, auf die jeweilige Hardware ab. In der aktuellen Version unterstützen die Agenten die Symbian und Windows Mobile Plattform.

2.2.2 Anwendungsszenario

Abbildung 2.4 zeigt das grundlegende Anwendungsszenario von MobileTest. Der Ablauf lässt sich wie folgt gliedern:

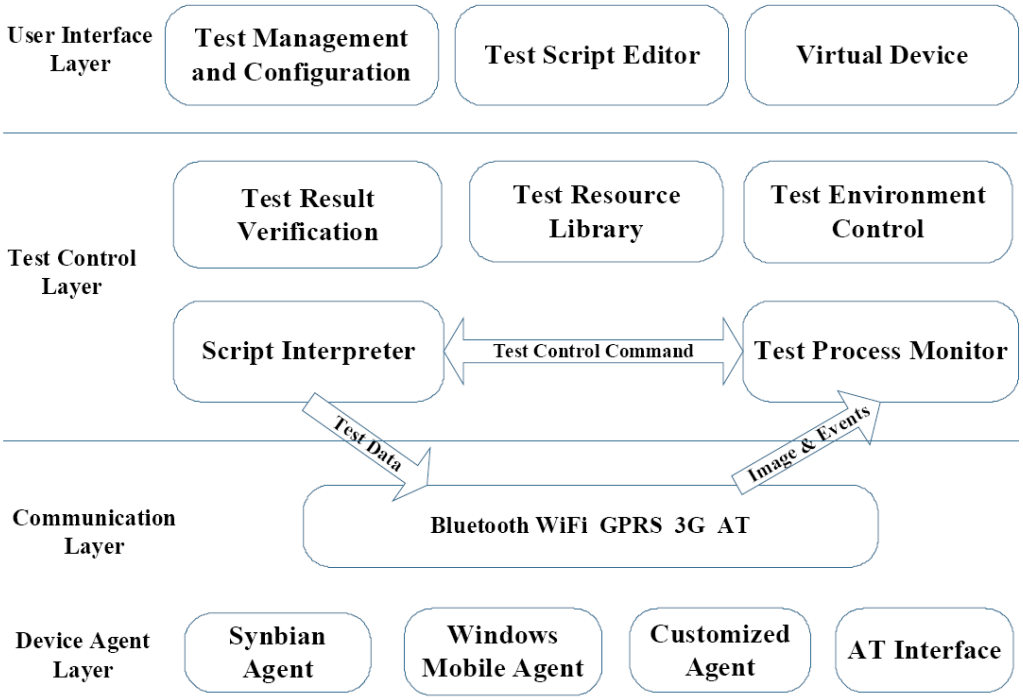

Abbildung 2.3: MobileTest Architektur (Bo u. a., 2007)

1. Der Tester konfiguriert im Modul Test management and configuration das mobile Zielgerät und dessen Umgebung.
2. Der Tester erstellt Testszenarien und Testskripte, entweder indem er diese manuell im Script Editor schreibt oder er lässt diese vom Modul Virtual Device generieren. Dieses Modul bietet für gängige mobile Zielgeräte vorgefertigte Standard–Testfälle.
3. Die Testskripte werden zur Ausführung freigegeben.
4. Der Script Interpreter interpretiert die Skripte und erzeugt die entsprechenden, simulierten Eingaben, die über die Communication Layer an den Device Agent geschickt werden. Danach beendet er sich selber.
5. Der Device Agent simuliert die Eingaben auf dem Zielgerät und sendet Ergebnisse (als Screenshot oder sensitive event) an den Test Process Monitor.
6. Nachdem der Test Process Monitor die Ergebnisse des Testfalls erhalten hat, aktiviert er wieder den Script Interpreter. Dieser kann nun den nächsten Testfall verarbeiten. Anschließend speichert der Test Process Monitor das Ergebnis in der Test Resource Library.
7. Das Ergebnis des Testfalls kann nun ausgewertet werden (in Abb. 2.4 nicht abgebildet).

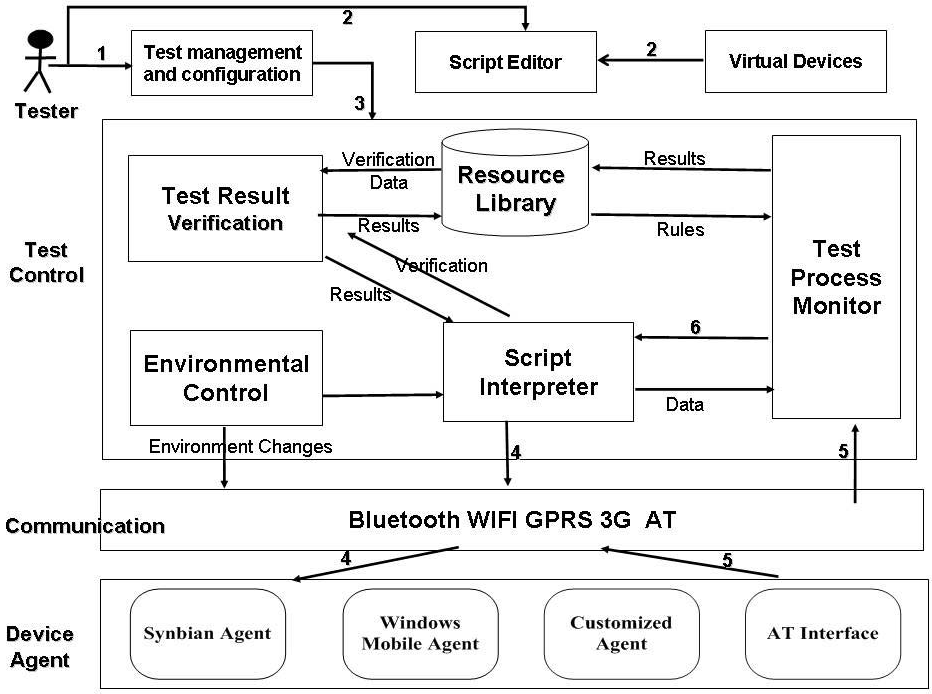

Abbildung 2.4: MobileTest Szenario (Bo u. a., 2007)

2.3 White Box Test: JaBUTi/ME

Aktuell gibt es zwar einige Ansätze um Black Box Tests bei mobilen Anwendungen durchzuführen, aber es existieren kaum Ansätze um die Korrektheit einer mobilen Anwendung zu testen. Natürlich sollte es aber auch möglich sein die Korrektheit von mobilen Anwendungen zu testen.
JaBUTi/ME (Delamaro u. a., 2006) ist ein Programm für solche White Box Tests auf mobilen Geräten. Im Gegensatz zu MobileTest unterstützt JaBUTi/ME keine unterschiedlichen Plattformen, sondern nur die J2ME (Java) Plattform. JaBUTi/ME überprüft die Korrektheit einer Anwendung, indem der Programmcode getestet wird (Coverage Testing). Die Teststrukturen leiten sich dabei, wie bei White Box Tests üblich, aus der Implementierung ab (Entscheidungen, Datenfluss, Interaktionen, usw.). Getestet werden die Anwendungen auf den mobilen Zielgeräten und nicht im Emulator.

JaBUTi/ME analysiert und testet nicht den Quellcode einer Anwendung, sondern den Java Byte–Code. Auf vielen mobilen Geräten ist nur der Byte–Code einer Anwendung verfügbar. Daher ist JaBUTi/ME in der Lage eine Anwendung direkt auf dem Zielgerät zu testen und ist nicht nur auf das Testen im Emulator beschränkt.

2.3.1 Strukturelles Testen auf mobilen Geräten

Strukturelles Testen basiert auf der internen Struktur der Implementierung einer Anwendung (White Box). Aus dieser Struktur werden die Testfälle abgeleitet. Beim strukturellen Testen werden sogenannte Control Flow Graphen als abstrakte Darstellung eines Programmes bzw. eines Programmteils erzeugt.

Control Flow Graph

Ein gerichteter Graph, bei dem die Knoten unteilbare Code–Blöcke repräsentieren. Die Kanten repräsentieren die möglichen Ablaufsequenzen zwischen den Blöcken. Aus einem solchen Graphen lassen sich relativ einfach Testfälle ableiten.
Beispiel: Jeder Knoten und jede Kante soll mindestens einmal traversiert werden. Das würde bedeuten, dass jede Zeile im Programmcode mindestens einmal ausgeführt wird.

Analyse

Zum strukturellen Testen führt JaBUTi/ME die folgenden Schritte aus.

1. Der Programmcode, der zu testenden Anwendung wird analysiert. Nach der Analyse wird aus dem Programmcode ein Control Flow Graph erzeugt.
2. Die Test–Anforderungen, die erfüllt sein müssen, werden generiert.
3. Der Programmcode wird instrumentalisiert. Bei der Instrumentalisierung des Programmcodes wird dieser um Funktionen zur Testauswertung erweitert. Diese Funktionen beinhalten Informationen welcher Block ausgeführt wird sowie Methoden, welche die Testergebnisse sammeln und speichern. Die Ausführung des instrumentalisierten Programms erzeugt, neben dem ursprünglichen Verhalten, eine Liste mit allen Punkten im Programm, die durch Ausführung des Testfalls erreicht wurden.
4. Der letzte Schritt ist die Analyse, welche Testfälle erfüllt sind.

2.3.2 Server-based testing

Wie bereits festgestellt wurde, ist es wichtig dass mobile Anwendungen auf dem Zielgerät und nicht nur im Emulator getestet werden. Ebenfalls wurde bereits festgestellt, dass die beschränkten Hardware–Ressourcen der mobilen Geräte das Testen auf dem Zielgerät erschweren. Es ist daher nicht praktikabel eine Testumgebung, wie JaBUTi/ME direkt auf dem mobilen Gerät auszuführen.

Um dieses Problem zu lösen verfolgt JaBUTi/ME einen Ansatz, der als „server-based testing“ bezeichnet wird (vgl. Abb. 2.5). Der instrumentalisierte Programmcode wird über ein beliebiges Netzwerk auf die mobilen Geräte übertragen. Der Programmcode wird auf den Zielgeräten ausgeführt und es werden die Ausführungsdaten des Tests über das Netzwerk versendet. Die Instrumentalisierung des Programmcodes erfolgt auf einem Standard–Computer mit installierten JaBUTi/ME. Dieser Computer fungiert als Server, sendet den Programmcode an die Zielgeräte und empfängt die Ausführungsdaten. Eine Auswertung der Daten kann ggf. ebenfalls auf diesem Computer erfolgen.
Dieser Ansatz ermöglicht eine Ausführung der Testfälle auf dem Zielgerät, ohne dass dessen beschränkte Hardware–Ressourcen ein Problem darstellen.

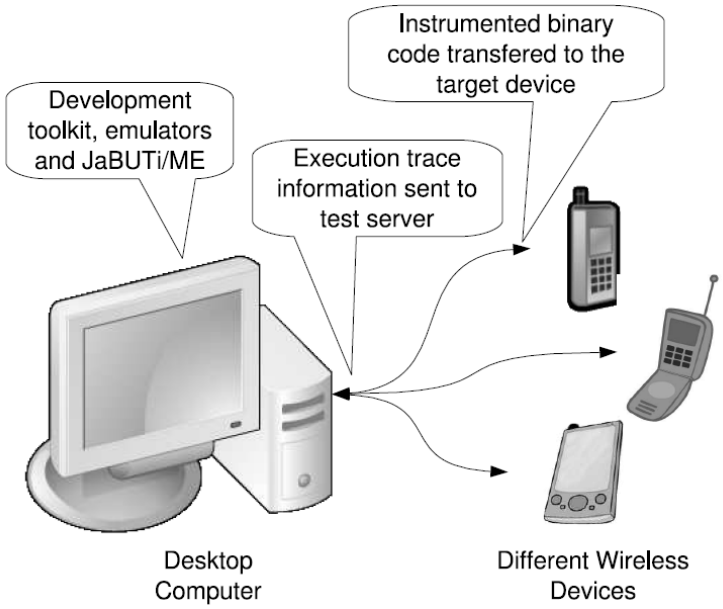

Abbildung 2.5: Server-based testing (Delamaro u. a., 2006)

3 Usability von mobilen Anwendungen

Das Entwerfen von effektiven und benutzbaren Benutzerschnittstellen für mobile Anwendungen, wird mit der steigenden Verbreitung von mobilen Geräten immer wichtiger. Allerdings sind Usability–Untersuchungen über die Interaktion von Benutzern mit mobilen Geräten schwierig. Viele traditionelle Usability–Labore sind optimiert für das Untersuchen von immobilen Anwendungen. Daher ist die verfügbare Infrastruktur meist nicht kompatibel mit den besonderen, gerätespezifischen Einschränkungen von mobilen Geräten.

3.1 Herausforderungen

Bei dem Aufbau und der Gestaltung von speziellen Usability–Untersuchungen für mobile Anwendungen treten eine Reihe von Herausforderungen auf (Schusteritsch u. a. (2007)):

Infrastrukturelle Herausforderungen

Die benötigte Zeit für den Aufbau und die Konfiguration der mobilen Geräte ist höher, als bei stationären Geräten, u. a. bedingt durch die schwierigere Bedienung der mobilen Geräte. Ebenso ist es problematisch den kleinen Bildschirm eines mobilen Gerätes mit mehreren Personen zu beobachten, bzw. den Bildschirm für eine spätere Auswertung abzufilmen. Die meisten Ansätze zur Lösung dieses Problems montieren kleine Kameras auf den mobilen Geräten, die den Bildschirm und ggf. den Benutzer filmen (siehe Abb. 3.1 und Abb. 3.2).
Wie eingangs bereits erwähnt, gibt es außerdem häufig Schwierigkeiten die mobilen Geräte in die vorhandene Infrastruktur einzubinden.

Herausforderungen durch die Benutzung von mobilen Geräten

Werden Kameras oder andere Geräte zur Überwachung der Untersuchung auf den mobilen Geräten montiert, kann dieses zu einer Veränderung in deren Benutzerbarkeit führen. Dadurch käme es dann zu einer Verfälschung der Untersuchungsergebnisse. Aus Abb. 3.1 und 3.2 ist ersichtlich, dass die Mobiltelefone mit den montierten Kameras z. B. zum Telefonieren

nicht mehr an das Ohr gehalten werden können.
Ein weiterer Punkt ist die soziale Komponente durch den mobilen Einsatz. Ein mobiles Gerät bzw. eine mobile Anwendung wird meistens in der Öffentlichkeit benutzt (vgl. Marti (2002)), d. h. das Verhalten des Benutzer wird durch die Umwelt beeinflusst. Ein weiterer Aspekt, der bei einer Usability–Untersuchung beachtet werden sollte.

Herausforderung Usability–Untersuchung für mobile Geräte

Beachtet werden sollte außerdem, dass die Benutzung eines mobilen Gerätes (kleiner Bildschirm und kleine Tastatur) für einen Benutzer deutlich anstrengender ist, als die Benutzung eines stationären Gerätes. Daher sollten die Sitzungen einer Usability–Untersuchung für mobile Geräte kürzer sein, als die einer normalen Usability–Untersuchung.

3.2 Feld- und Labortests

Bei einer Usability–Untersuchung für mobile Anwendungen stellt sich die Frage, ob diese als Feld- oder Labortest durchgeführt werden sollte (vgl. Duh u. a. (2006) und Nielsen u. a. (2006))?
Konventionelle Usabiliy–Untersuchen findet meistens im Labor statt. Mobile Anwendungen werden aber fast immer „unterwegs“, also im Feld benutzt.

Für einen Labortest spricht, dass dieser weniger zeitaufwändig ist. Die bereits vorhandene Infrastruktur kann meist, wenn auch ggf. mit einigen Einschränkungen, verwendet werden. Dieses erleichtert das Speichern und Sammeln der gewonnenen Daten. Die Infrastruktur erleichtert außerdem die Kontrolle der Untersuchung.

Für einen Feldtest spricht, dass dort unkontrollierbare Umweltfakoren auftreten, die das Benutzerverhalten beeinflussen und in einem Labortest nicht nachempfunden werden können. Das führt dazu, dass im Feldtest signifikant mehr Usability–Fehler gefunden werden und die Benutzer für die Aufgaben deutlich länger brauchen (Nielsen u. a., 2006).

Ein Feldtest ist zwar wesentlich aufwändiger zu gestalten und zu kontrollieren, als ein Labortest, aber nur im Feldtest können alle Usability–Fehler, die in der realen Anwendungssituation auftreten können auch gefunden werden. Daher sollten Usability–Untersuchungen wenn möglich, im Feld vorgenommen werden.

Abbildung 3.1: Handy mit Mini-Kamera und Barcode-Scanner (Nielsen u. a., 2006)

Abbildung 3.2: Handy mit Observation-System (Schusteritsch u. a., 2007)

4 Zusammenfassung und Ausblick

4.1 Zusammenfassung und Fazit

4.1.1 Testen von mobilen Anwendungen

Das Testen von mobilen Anwendungen ist auf konzeptioneller Ebene weitreichend erforscht. Es existieren viele interessante Ansätze, von denen mit MobileTest (Black Box Testing) und JaBUTi/Me (White Box Testing) zwei in dieser Arbeit vorgestellt wurden. MobileTest und JaBUTi/Me sind zwei Programme, die die grundlegenden Schwierigkeiten beim Testen von mobilen Anwendungen lösen. Dabei verfolgen sie zwei unterschiedliche Ansätze; MobileTest erzeugt wartbare und wiederverwendbare Black Box Tests für verschiedene, mobile Geräte. Mit JaBUTi/Me ist es möglich die Korrektheit der Implementierung einer Anwendung auf mobilen Geräten zu testen (White Box Test). Zum Testen einer mobilen Anwendung sollte sowohl die Funktionalität (Black Box Test), als auch die Korrektheit der Implementierung (White Box Test) getestet werden. Daher ist es sinnvoll sowohl MobileTest als auch JaBUTi/Me einzusetzen (oder entsprechende andere Programme, die hier nicht vorgestellt wurden). Allerdings fehlt es in diesem Bereich an einer Übersichtsarbeit, die die einzelnen Ansätze vergleicht und evaluiert. Dieses könnte für weiterführende Veranstaltungen im Rahmen des Studiums ein interessantes Thema sein.

4.1.2 Usability von mobilen Anwendungen

Im Bereich der Usability–Untersuchungen für mobile Anwendungen gibt es einige interessante Ansätze. Allerdings haben die meisten das Konzeptionsstadium noch nicht verlassen. Im Gegensatz zu Usabiliy–Untersuchungen für immobile Anwendungen, hat die Infrastruktur für mobile Anwendungen das Experimentierungsstadium noch nicht verlassen. Hier fehlt es noch an einer passenden Infrastruktur. In dieser Arbeit wurden die grundlegenden Herausforderungen von mobilen Usability–Untersuchungen identifiziert und es wurden einige Lösungsansätze skizziert. Außerdem wurde darauf eingegangen, warum Usability–Untersuchungen wenn möglich im Feld und nicht im Labor stattfinden sollten.
Der Bereich der Usability–Untersuchungen für mobile Anwendungen ist noch nicht komplett

erschlossen. Im Rahmen von weiterführenden Veranstaltungen könnten hier eigene Ansätze oder der Aufbau einer passenden Infrastruktur erarbeitet werden.

4.2 Ausblick

Im kommenden Semester werde ich an dem Projekt „Pervasive Gaming“ bei Prof. Dr. Olaf Zukunft teilnehmen. In diesem Projekt werden wir uns mit Anwendungen für mobile Geräte befassen. Daher werde ich meine, im Rahmen der Ausarbeitung und des Vortrages gewonnenen Erkenntnisse dort mit einfließen lassen können.
Das Testen von mobilen Anwendungen erscheint mir als Thema für eine Master–Thesis nicht interessant genug. Voraussichtlich wird meine Master–Thesis aber im Bereich der mobilen Anwendungen angesiedelt sein. In diesem Fall werde ich, die hier gewonnenen Erkenntnisse als Grundlage in meine Master–Thesis mit einfließen lassen können.
Usability–Tests für mobile Anwendungen erscheint mir als Thema für eine Master–Thesis interessanter. Daher könnte ich mir vorstellen, dieses Thema in weiteren Veranstaltungen noch zu vertiefen.

Glossar

Deployment Aus dem Englischen (Auslieferung). In diesem Kontext bezeichnet es die Verteilung und Installation von Software auf Zielsystemen.

Emulator Ein Emulator ist ein System, das ein anderes nachahmt. Ein Software-Emulator ist eine Software, die ein System nachahmt. Beispiel: Ein (Software–)Emulator, der ein mobiles Gerät nachahmt.

Feldtest Test eines Produktes unter realen Einsatzbedingungen.

HAW Hochschule für Angewandte Wissenschaften.

J2ME Java Micro Edition; ist eine Sammlung von Java-Programmierschnittstellen für mobile Geräte.

Java Eine von Sun Microsystems entwickelte, objektorientierte Programmiersprache.

Java Byte–Code Der kompilierte Java–Quellcode.

Labortest Test eines Produktes unter Laborbedingungen.

PDA Personal Digital Assistant: Kompakte, tragbare Computer, die hauptsächlich für persönliche Kalender–, Adress- und Aufgabenverwaltung genutzt werden.

Test In diesem Kontext Softwaretest: Stichprobenartige Überprüfung der Funktionalität. Testen ist ein Prozess mit der Absicht Fehler zu finden.

Testdaten Eingabedaten für einen Testfall.

Testfall Ein elementarer, funktionaler Softwaretest, der der Überprüfung einer, durch eine Spezifikation, zugesicherten Eigenschaft dient.

Testskript Eine Aneinanderreihung von Testfällen. Dienst der automatischen, sequentiellen Ausführung von mehreren Testfällen.

Testszenario Bezeichnet eine Zusammenstellung von Testfällen.

Usability Bezeichnet in der Informatik die Gebrauchstauglichkeit bzw. die Benutzbarkeit eines Produktes. Bei Software–Produkten steht häufig die Benutzerfreundlichkeit im Vordergrund.

Literaturverzeichnis

[Bach 2003] BACH, James: *Exploratory Testing Explained*. http://www.satisfice.com/articles/et-article.pdf (Stand 01.05.2008). 04 2003

[Binder und Hanlon 2005] BINDER, Robert V. ; HANLON, James E.: The advanced mobile application testing environment. In: *SIGSOFT Softw. Eng. Notes* 30 (2005), Nr. 4, S. 1–1. – ISSN 0163-5948

[Bo u. a. 2007] BO, Jiang ; XIANG, Long ; XIAOPENG, Gao: MobileTest: A Tool Supporting Automatic Black Box Test for Software on Smart Mobile Devices. In: *AST '07: Proceedings of the Second International Workshop on Automation of Software Test*. Washington, DC, USA : IEEE Computer Society, 2007, S. 8. – ISBN 0-7695-2971-2

[Cundy 2001] CUNDY, M.: *Testing Mobile Applications is Different from Testing Traditional Applications*. Veritest Tester's Network. 08 2001

[Dalal u. a. 1999] DALAL, S. R. ; JAIN, A. ; KARUNANITHI, N. ; LEATON, J. M. ; LOTT, C. M. ; PATTON, G. C. ; HOROWITZ, B. M.: Model-based testing in practice. In: *ICSE '99: Proceedings of the 21st international conference on Software engineering*. Los Alamitos, CA, USA : IEEE Computer Society Press, 1999, S. 285–294. – ISBN 1-58113-074-0

[Delamaro u. a. 2006] DELAMARO, M. E. ; VINCENZI, A. M. R. ; MALDONADO, J. C.: A strategy to perform coverage testing of mobile applications. In: *AST '06: Proceedings of the 2006 international workshop on Automation of software test*. New York, NY, USA : ACM, 2006, S. 118–124. – ISBN 1-59593-408-1

[Duh u. a. 2006] DUH, Henry Been-Lirn ; TAN, Gerald C. B. ; CHEN, Vivian H. 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

[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 20th conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction: design: activities, artefacts and environments*. New York, NY, USA : ACM, 2006, S. 261–268. – ISBN 1-59593-545-2

[Hartman u. a. 2007] HARTMAN, Alan ; KATARA, Mika ; PARADKAR, Amit: Domain specific approaches to software test automation. In: *ESEC-FSE '07: Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*. New York, NY, USA : ACM, 2007, S. 621–622. – ISBN 978-1-59593-811-4

[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 Computing* 10 (2006), Nr. 6, S. 345–355. – ISSN 1617-4909

[Lee und Grice 2004] LEE, Kwang B. ; GRICE, Roger A.: Developing a New Usability Testing Method for Mobile Devices. In: *Professional Communication Conference, 2004. IPCC 2004. Proceedings. International*, 2004, S. 115–127

[Mahmoud 2002] MAHMOUD, Qusay H.: *Testing Wireless Java Applications*. Sun Developer Network http://developers.sun.com/mobility/midp/articles/test (Stand: 01.05.2008). 11 2002

[Marti 2002] MARTI, Stefan: *How does the user interface design of mobile devices influence the social impact of mobile communication?* MIT Media Lab http://web.media.mit.edu/stefanm/generals/mainpaper_social_impact.2002.02.18.pdf (Stand 01.05.2008). 02 2002

[do Nascimento und Machado 2007] NASCIMENTO, Laisa H. O. do ; MACHADO, Patricia D. L.: An experimental evaluation of approaches to feature testing in the mobile phone applications domain. In: *DOSTA '07: Workshop on Domain specific approaches to software test automation*. New York, NY, USA : ACM, 2007, S. 27–33. – ISBN 978-1-59593-726-1

[Nielsen u. a. 2006] NIELSEN, Christian M. ; OVERGAARD, Michael ; PEDERSEN, Michael B. ; STAGE, Jan ; STENILD, Sigge: It's worth the hassle!: the added value of evaluating the usability of mobile systems in the field. In: *NordiCHI '06: Proceedings of the 4th Nordic conference on Human-computer interaction*. New York, NY, USA : ACM, 2006, S. 272–280. – ISBN 1-59593-325-5

[Nokia 2004] NOKIA: Series 60 Developer Platform 2.0: Guidelines For Testing J2ME Applications / Nokia Corporation. 03 2004. – Forschungsbericht

[Satoh 2003] SATOH, Ichiro: A Testing Framework for Mobile Computing Software. In: *IEEE Transactions on Software Engineering* 29 (2003), Dezember, Nr. 12, S. 1112–1121

[Satoh 2004] SATOH, Ichiro: Software testing for wireless mobile computing. In: *IEEE Wireless Communications* 11 (2004), 10, Nr. 5, S. 58–64

[Schultz 2006] SCHULTZ, David: 10 usability tips & tricks for testing mobile applications. In: *interactions* 13 (2006), Nr. 6, S. 14–15. – ISSN 1072-5520

[Schusteritsch u. a. 2007] SCHUSTERITSCH, Rudy ; WEI, Carolyn Y. ; LAROSA, Mark: Towards the perfect infrastructure for usability testing on mobile devices. In: *CHI '07: CHI '07 extended abstracts on Human factors in computing systems*. New York, NY, USA : ACM, 2007, S. 1839–1844. – ISBN 978-1-59593-642-4

[Srirama u. a. 2006] SRIRAMA, Satish ; KAKUMANI, Rajasekhar ; AGGARWAL, Akshai ; PAWAR, Pravin: *Effective Testing Principles for the Mobile Data Services Applications*. IEEE. 2006