Engineering Mobile User Experience: Think. Design. Fail. Iterate. Publish.

Markus Heckner[1], Tim Schneidermeier[1], Alexander Bazo[2], Thomas Wagner[1], Thomas Wilhelm[1], Christian Wolff[1]

Lehrstuhl für Medieninformatik, Universität Regensburg[1]
Lehrstuhl für Informationswissenschaft, Universität Regensburg[2]

Zusammenfassung

Der Vertrieb von Smartphone-Applikationen über zentralisierte Marktplätze wie *Android Market*, *Apple iTunes* oder *Nokia Ovi Store* ist heute fest etabliert. Dieser Vertriebsweg bietet Chancen, aber auch Risiken für die Entwickler: Die Nutzer erhalten einen direkten Feedbackkanal, auf dem sie Applikationen für alle anderen Nutzer sichtbar bewerten und kommentieren können. *User Experience* ist hier ein Schlüssel zum Erfolg einer App. Gleichzeitig erfordert die starke Konkurrenz durch die hohe Anzahl von Entwicklern ständig neue Produkte, um sich von bestehenden Angeboten abzuheben. In diesem Praxisbericht wird ein Vorgehensmodell vorgeschlagen, das den Nutzer ins Zentrum jeder mobilen Anwendungsentwicklung stellt, mit dem Ziel, effizient Software zu entwickeln, die tatsächliche Probleme der Nutzer löst und sich durch positive *User Experience* auszeichnet. Anhand eines konkreten Projekts wird der erfolgreiche Einsatz des Modells dokumentiert und abschließend diskutiert.

1 Einleitung

Bis vor wenigen Jahren spielten mobile Applikationen eine untergeordnete Rolle. Die Ursachen sind vielfältig, sind aber u.a. in mühsamen Zahlungswegen und großen Hürden bei der Beschaffung begründet. Beispielsweise gestaltete sich der Erwerb einer Applikation für die Windows Mobile-Plattform als mühsam: Nutzer mussten mit Hilfe eines PCs nach einer geeigneten Applikation im Web suchen, selbst zur Website des Anbieters navigieren und die Bezahlung nach Einrichtung eines Benutzerkontos vornehmen. Die eigentliche Installation auf dem Gerät erfolgte nach dem Download mit *ActiveSync* oder vergleichbaren Produkten[1].

Apple, Google, Nokia, Microsoft und RIM vereinfachen diesen Prozess und bieten mittlerweile jeweils einen zentralisierten Marktplatz für mobile Anwendungen an, die den Beschaf-

[1] Ein Beispiel für diesen aufwändigen Prozess findet sich auf: http://pocketpccentral.net/help/softinstall.htm.

fungsprozess auf einen Klick reduzieren: Nach einer einmaligen Registrierung unter Angabe der Zahlungsdaten können Anwendungen direkt auf dem Gerät gesucht und mit einem „Tap“ der Kauf und die Installation bestätigt werden. Die Einfachheit dieses Prozesses wird auch für die Anwendungen selbst gefordert: *User Experience* wird zunehmend als Muss angesehen, um Produkte und Dienstleistungen erfolgreich zu vertreiben (Accenture Technology Labs 2011). Für alle diese Marktplätze liegen umfangreiche Dokumentationen vor und öffnen durch geringe Hürden den Markt für eine Vielzahl von professionellen Entwicklern und Hobbyprogrammierern (Apple Inc. 2011a).

Dieser Beitrag schlägt ein effizientes Modell für die Entwicklung mobiler Applikationen vor, das die dabei auftretenden spezifischen Herausforderungen berücksichtigt: Zunächst werden Erfolgsfaktoren, die sich an die mobile Applikationsentwicklung ergeben, dargestellt (Kap. 2). Ausgehend von der Ideenfindung wird anschließend ein nutzerzentrierter Entwicklungsprozess beschrieben, der bis zur Veröffentlichung der App reicht (Kap. 3). Der Einsatz des vorgeschlagenen Modells wird abschließend anhand einer Fallstudie über die Entwicklung einer mobilen App zur Steuerung von Präsentationen veranschaulicht. Das Modell und die App entstanden während der Vorbereitung auf einen Android-Kurs an der Universität Regensburg im Studiengang Medieninformatik. Ziel dieses Kurses ist der Erwerb allgemeiner Grundlagen der Anwendungsentwicklung am Beispiel Android. Ein weiteres Lehrziel ist die Verbindung von Programmiertechniken mit Methoden des Usability Engineerings.

2 Erfolgsfaktoren für die Entwicklung auf Android Market, App Store & Co.

Der Wandel vom individuellen Vertrieb zu einer zentralisierten Verkaufsplattform erschließt ein großes Umsatzpotential für mobile Apps. Die folgenden Konsequenzen lassen sich aufgrund der Besonderheiten des Vertriebskanals erkennen:

- Erste User-Reviews bestimmen über zukünftige Verkäufe („make it or break it“).
- Signifikante Umsätze werden aufgrund des meist geringen Produktpreises nur durch hohe Stückzahlen erreicht.
- Die Hürde für Alternativkäufe ist aufgrund des geringen Preises niedrig.
- Entwickler sind auf Empfehlungs- und virales Marketing infolge hoher Konkurrenz durch Parallelentwicklungen und der immensen Anzahl an Apps angewiesen.

Hieraus lassen sich die nachfolgend beschriebenen Erfolgsfaktoren herausarbeiten.

2.1 Bedarf

Jede Entwicklung sollte den Nutzer von Anfang an mit in die Überlegungen einbeziehen: Es muss identifiziert werden, welche Probleme und Bedürfnisse eine App lösen kann, bevor mit der eigentlichen Softwareentwicklung begonnen wird. Eine technisch fehlerfreie App, die

niemand braucht, kann von Anfang an keinen Erfolg haben. Die tatsächliche Brauchbarkeit der angebotenen Apps schwankt: Viele Apps können viel, lösen aber nicht zwingend Probleme der Nutzer. Auch auf den ersten Blick sinnlose, aber erfolgreiche Scherz-Apps treffen immer auf ein Bedürfnis der Nutzer, und sei es auch nur der Wunsch nach Unterhaltung oder Außendarstellung.

2.2 Mobile User Experience

Usability Engineering und *User Experience Design* (UXD) gewinnen gerade in den letzten Jahren zunehmend an Bedeutung und durchdringen mittlerweile viele unterschiedliche Märkte und Lebensbereiche. Egal ob Software, Weckuhr oder Webseite – dem Benutzer soll ein bestmögliches Nutzungserlebnis bereitet werden (Klauser & Walker 2007; Norman 1988). Um dies zu erreichen, gilt es neben den (funktionellen) Anforderungen der Gebrauchstauglichkeit auch alle anderen Aspekte zu berücksichtigen, die die Wahrnehmung des interaktiven Systemen als Ganzes beeinflussen (Ästhetik, Emotionen, Joy of Use, (z.B. Alben 1996)).

Mobile Applikationen unterliegen neben allgemein gültigen Gestaltungsrichtlinien für Software (z.B. Shneiderman & Plaisant 2009) weiteren Herausforderungen (z.B. Apple Inc. 2011b; Ginsburg 2010; Google 2011), die für ein benutzerfreundliches Produkt im Sinne der *Usability* und *User Experience* (UX) berücksichtigt werden müssen: Unterschiedliche Hardware-Ressourcen, (vergleichsweise) kleine Displays, variierende Interaktionstechniken und der mobile Nutzungskontext bedürfen zusätzlicher Beachtung. Ein wesentlicher Faktor guter Bedienbarkeit mobiler Software ist die konsequente Erfüllung der Erwartung des Benutzers an Erscheinung und Benutzung. Ein konsistenter Übergang vom genuinen *User Interface* des Betriebssystems des Geräts zum *Interface* der eigenen App ermöglicht dem Nutzer eine intuitive Bedienung und soll ein positives Benutzererlebnis erzeugen. Die Interaktion innerhalb einer Applikation sollte auf solchen Prinzipien beruhen, die der Benutzer bereits aus dem Kontext des übergeordneten Systems bzw. von anderen Applikationen kennt. Seitens der Hersteller dieser Systeme besteht der Wunsch, durch die Verbreitung und Kommunikation bestimmter *User Interface Design Patterns* und *Guidelines* diese möglichst konsistente Erscheinung und Bedienbarkeit innerhalb der heterogenen Menge an Applikationen zu gewährleisten (Apple Inc. 2011b). Diese Dokumente sind aber häufig sehr umfangreich (vgl. z.B. RIM mit über 100 Seiten und Apple mit derzeit über 150 Seiten (Apple Inc. 2011b; Research in Motion Limited 2010), und werden somit mutmaßlich nur selten vollständig von den Entwicklern erfasst. Zusätzlich lösen diese Guidelines nicht das Transferproblem: Auch nachdem die Dokumente erarbeitet wurden, muss ein konkretes Problem im Benutzerinterface der Applikation umgesetzt werden. Hierbei werden die Entwickler nicht von den Guidelines unterstützt.

2.3 Effiziente Entwicklung

Zur Applikationsentwicklung sind nur geringe Hürden zu überwinden: Entwicklungsumgebungen und Plugins sind meist kostenlos, die Teilnahme am Entwicklerprogramm kostet häufig weniger als $ 100,- und die Frameworks sind auf einfache Entwicklung hin optimiert. Diese Faktoren führen zu einer hohen Anzahl von Entwicklern. Beobachtet man den Markt

genauer, werden innovative Apps schnell kopiert und es existieren häufig mehrere konkurrierende Angebote. Dies erfordert einen schnellen Entwicklungsprozess, um sich am Markt zu etablieren und die Risiken für Fehlentwicklungen zu minimieren. Derzeit (Stand Juni 2011) sind mehrere Hunderttausend Apps auf dem *Android Market* verfügbar (Wikipedia 2011).[2]

2.4 Codequalität

Die Marktplattform erzeugt ein transparentes Meinungsbild über die Applikationen: Reviews werden unmittelbar veröffentlicht, und der Entwickler hat keine Möglichkeit, diese zu löschen oder Stellung zu nehmen. Abstürze und Bugs sind ein sicherer Weg zu schnellen negativen Feedbacks: Jeder Absturz erzeugt potentiell negative Reaktionen, die Nutzer bereitwillig als Kommentare äußern. Google versucht diesem Problem durch explizite Guidelines zur Konzipierung der Applikation auf Codeebene entgegenzutreten: Es werden *best practices* formuliert und *Design Patterns* ausgegeben. Beispielweise ergibt sich durch das Design der Plattform, das *model-view-controller*-Prinzip (MVC) für die Entwicklung aufzugreifen. Die Trennung von UI und Programmlogik ist durch die Architektur der Plattform vorgegeben. Weitere Patterns betreffen die lose Kopplung (*loose coupling*) einzelner Komponenten und die späte Bindung (*late binding*) zur Laufzeit.

2.5 Fazit

Die oben beschriebenen Erfolgsfaktoren lassen sich nach unseren Erkenntnissen wie folgt zusammenfassen (vgl. Abbildung 1):

- Der Nutzer muss im Zentrum der Entwicklung stehen (*Bedarf*)
- Nur positive User Experience kann langfristig Erfolg sichern
- Die hohe Innovationsgeschwindigkeit erfordert effiziente Entwicklungspraxis
- Hochwertiger Code kann Frustration und schlechte Bewertungen verhindern helfen

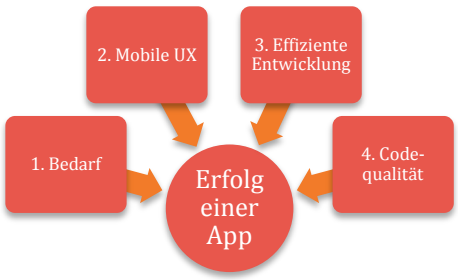

Abbildung 1: Fazit: Erfolgsfaktoren einer App

[2] Die Informationsplattform *AndroLib* geht für den Android-Markt von knapp 400.000 Apps und Spielen bei etwa fünf Milliarden Downloads aus, vgl. http://www.androlib.com/appstats.aspx [Zugriff Juni 2011].

3 Think. Design. Fail. Iterate. Publish.

Basierend auf den Erfolgsfaktoren aus den vorhergehenden Abschnitten schlagen wir ein Vorgehensmodell für die mobile Applikationsentwicklung vor, das von der Idee bis zur Veröffentlichung der Applikation reicht (vgl. Abbildung 2) und das mit leichten Anpassungen für die Entwicklung eigener Apps eingesetzt werden kann. Der Fokus liegt weniger auf der Produktion formaler Dokumente, sondern auf praxis- und ergebnisorientierter Entwicklung.

Viele Softwareprojekte scheitern an mangelhaft formulierten Anforderungen (Selby 2007). *Usability Engineering Frameworks* erkennen dieses Problem, setzen an diesem Punkt an und beziehen den Nutzer so früh wie möglich in den Entwicklungsprozess mit ein (vgl. z.B. Dahm 2008, 23). Unterschiedlichen Modellen ist die Konzentration auf die Nutzer und deren Anforderungen sowie die iterative Entwicklung eines Systems gemein: Es wird von Anfang an akzeptiert, dass ein perfekter Entwurf der Benutzeroberfläche nicht in einem Schritt möglich ist. Voraussetzung für die erfolgreiche Entwicklung ist, dass das Entwicklungs- und Designteam verinnerlicht haben muss, dass Scheitern ein wichtiger Teil des Entwicklungsprozesses ist, und jede Kritik eine Chance darstellt, ein optimales Produkt zu entwickeln.

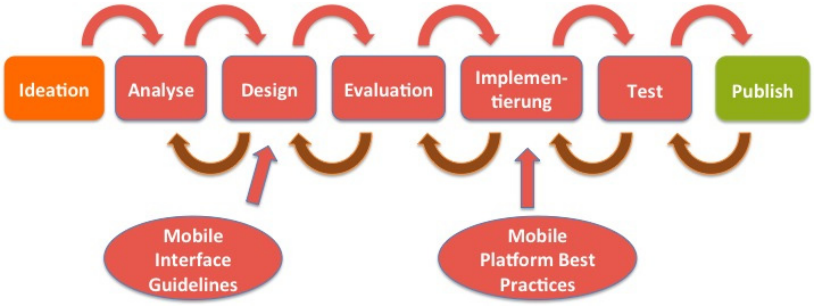

Abbildung 2: Mobile Usability Engineering Framework

Die Phasen sind durch Rücksprünge auch mit den Vorgängerphasen verbunden, eine Evaluationsphase verdeutlicht den Unterschied zu klassischen Modellen wie dem Wasserfallmodell im Software Engineering (Royce 1987): Ab der Designphase lässt jede Phase Rücksprünge zu, auch über mehrere Phasen hinweg. Vor dem eigentlichen Usability Engineering steht die Idee. In der Ideenfindungsphase (*Ideation*) erweisen sich unterschiedliche Methoden eines *Design Thinking*-Prozesses als effizientes Mittel zur Identifikation vielversprechender Ideen (Brainstorming, Beobachtungsstudien, etc.). Die Entwicklung sollte nicht vom Entwickler ausgehen („Was können wir?“), sondern von Beginn an auf die zukünftigen Nutzer ausgerichtet sein („Was brauchen die Nutzer?“). Die Erkenntnisse aus der Ideenfindungsphase fließen direkt in die Analyse ein und werden durch eine technische Analyse (z.B. als *Proof of Concept*) ergänzt. Sind Inhalt und grobe Zielrichtung des Projekts abgesteckt, werden die Apps erarbeitet. Für die funktionale Ausarbeitung des Konzepts eignen sich *Sketching* (schnelle Skizzen, vgl. (Craft & Cairns 2009) zum ersten Entwurf der Screens und *Paper Prototyping* (Snyder 2003) zur Evaluation des Applikationsverlaufs (Warfel 2009). Ziel ist

die Ausarbeitung der Ideen und die sofortige Validierung durch das Team und Nutzer. Während der gemeinsamen Diskussion entsteht häufig erst der Mehrwert der App. In der Designphase werden bestehende UI-Richtlinien (Apple Inc. 2011b; Google 2011) beachtet und umgesetzt, um eine erwartungskonforme Darstellung zu erreichen. Der Fokus liegt aber auf der Erarbeitung und Evaluation der Ideen durch Einholen von Feedback. Während der Implementierung werden laufend neue Softwarestände generiert, die dann mithilfe informeller Usability-Tests weiter überprüft und ggf. angepasst werden können. Nach abgeschlossener Implementierung folgt die Testphase, welche alle Abläufe auf technische Korrektheit prüft. Diese abschließende Evaluation ermöglicht auch die Gesamtschau auf die Software. Erst jetzt können alle Komponenten im Zusammenspiel getestet werden. Nach erfolgreichen Tests erfolgt die Veröffentlichung der App. Auch nach Veröffentlichung lassen sich durch weitere Tests bzw. Kommentare der Nutzer gewonnene Erkenntnisse in die App integrieren und durch Updates einspielen.

4 Fallstudie: Entwicklung eines Android-Presenters

Dieser Abschnitt beschreibt die konkrete Anwendung des Modells am Beispiel einer Presenter-Software für Android-basierte Geräte, die über Bluetooth mit einem Rechner gekoppelt sind. Die App ermöglicht Nutzern, drahtlos die Folien ihrer Präsentationen durch Gestensteuerung auf dem Handy vor- und zurückzuschalten und ersetzt somit einen Hardware-Presenter oder die Benutzung der Tastatur des Computers. Es wurde bewusst auf Reduzierung und Aufgabenangemessenheit (i. S. der ISO 9241-110) der App gesetzt.

4.1 Ideation

Grundgedanke des Projekts war die Entwicklung einer App, die auf einen tatsächlichen Bedarf trifft. Die Ideenfindungsphase konzentrierte sich auf Methoden des *Design Thinking* (Meinel et al. 2009). Für eine erste Ideengenerierung wurde ein Brainstorming in zwei Durchgängen von je 10 Minuten durchgeführt. Das Ziel war es zunächst, sämtliche Ideen zu sammeln, die im Kontext mobiler Applikation einen Mehrwert für den Benutzer darstellen könnten. Die technische Machbarkeit bzw. die Wirtschaftlichkeit spielten in dieser Phase eine noch untergeordnete Rolle. Eine kurze Diskussion der Ergebnisse zwischen den beiden Phasen sollte die Ideenfindung zusätzlich anregen und die Erweiterung bestehender Ideen anregen. So konnten Ideen näher erläutert werden und Synergieeffekte für den zweiten Durchgang des Brainstormings gewonnen werden. In einer abschließenden Debatte einigten sich alle Teammitglieder auf die Umsetzung einer Idee. Im Anschluss wurde die Idee mehrfach in Form von *Elevator Pitches* (Gray et al. 2010) an potentiellen Nutzern evaluiert, um die Zielgruppe so früh als möglich in den Designprozess miteinzubeziehen. Die positive Resonanz zeigte den tatsächlichen Bedarf der Nutzer. Gleichzeitig konnte ein Bild des zukünftigen Nutzungskontexts gewonnen werden. Als Zielgruppe wurden der Bildungskontext (Universitäten, Schulen) sowie der Unternehmenskontext identifiziert.

4.2 Analyse

In der Analysephase wurde die technische Machbarkeit überprüft und eine Wettbewerbsanalyse ähnlicher Produkte durchgeführt.

4.2.1 Marktanalyse

Um weitere funktionale Anforderungen für die Applikation zu ermitteln, wurde der Android-Market auf vergleichbare Produkte hin untersucht. Dabei wurden vor allem der Verkaufspreis, vorhandene bzw. fehlende Features, *Usability* und technische Probleme betrachtet. Hierzu wurden die Konkurrenzprodukte kurzen informellen Evaluationen unterzogen und bestehende Reviews ausgewertet. Besonders letztere bestätigten die erarbeiteten und anhand kurzer Nutzerbefragungen überprüften Forderungen nach intuitiver Benutzbarkeit und technischer Einfachheit. Die Unterstützung aller verbreiteten Präsentationssoftwarepakete sowie die Unabhängigkeit vom Betriebssystem des Host-Computers konnten als weitere technische Anforderungen identifiziert werden. Die Überfrachtung mit Features und die Fehlerhaftigkeit bestehender Lösungen wurden als die größten Problemfelder ermittelt. Bei Bewältigung dieser Herausforderungen erschien dem Designteam die Platzierung der eigenen Software als innovativ und wirtschaftlich vielversprechend. Als Konsequenz dieser ersten Analyse sollte eine App entwickelt werden, die mit möglichst wenigen Features den Nutzerbedürfnissen und dem Nutzungskontext gerecht wird.

4.2.2 Technische Machbarkeit (Proof of Concept)

Ausgehend von der zur Verfügung stehenden soft- und hardwareseitigen Möglichkeiten zeitgemäßer Android-Smartphones (Matos & Grasser 2010) wurde die technische Machbarkeit einer Presenter-App untersucht. Zentraler Punkt war die Identifikation einer einfachen und weitverbreiteten Möglichkeit, Smartphone und Computer miteinander zu verbinden. Das Verwenden der Bluetooth-Technologie bot sich aufgrund der weiten Verbreitung und des relativ einfachen Pairing-Prozesses auf Nutzerseite an.

4.3 Design: Sketching

In einer ersten zehnminütigen Runde entwarf jeder der vier Teilnehmer selbstständig einige unterschiedliche mögliche Layouts. Da in dieser Phase das Ziel verfolgt wird, möglichst viele Ideen zu generieren, können auch Nicht-Experten im *UI Design* an diesem Workshop teilnehmen. Um die Entwürfe in realistische Bahnen zu lenken, wurde eine Palette von zulässigen Interface-Elementen für alle sichtbar als Referenz platziert. Anschließend präsentierten die Teilnehmer Ihre Entwürfe. Aus der Diskussion der Entwürfe entstand bei allen Mitwirkenden eine Vorstellung davon, wie die zukünftigen Screens aussehen könnten. Diese gemeinsame Vorstellung reduziert zukünftige Missverständnisse und Kommunikationsprobleme und gestaltet so Kommunikation innerhalb des Teams effizienter. Abbildung 3 zeigt das Resultat des *Sketching* für den Präsentationsbildschirm und das tatsächliche Endergebnis. Dieses ist um zwei Funktionen erweitert worden, welche sich in der begleitenden Evaluation als relevant herausgestellt hatten.

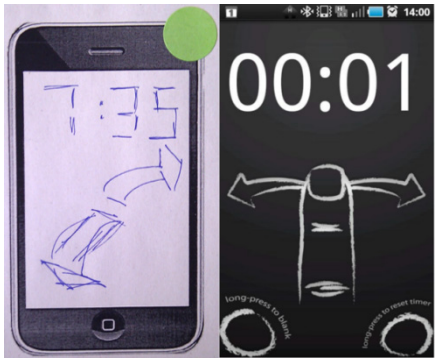

Abbildung 3: Ergebnis des Sketching (links) und Bildschirm bei Release (rechts)

4.4 Softwareentwicklung

Die Softwareentwicklung auf der Basis der Design-Entwürfe erfolgte mit Hilfe der für die Android 2-Plattform (Burnette 2010) verfügbaren Softwarewerkzeuge in einem kooperativen Arbeitsmodus innerhalb einer relativ kurzen Zeitspanne.

4.5 *Guerilla Usability Tests*

Ziel des Guerilla-Ansatzes ist eine möglichst kostengünstige und zeitsparende Evaluation der Software auf ihre technische Korrektheit sowie Identifikation der wichtigsten Usability-Probleme. Auf eine aufwendige Videoaufzeichnung und Auswertung wurde bewusst verzichtet: Zwei Beobachter protokollierten schriftlich auffällige Handlungen der Testpersonen sowie deren Verbesserungsvorschläge und Wünsche. Die aufgabenbasierten *Usability*-Tests wurden mit drei Versuchspersonen durchgeführt. (Krug 2009) stellt fest, dass bereits bei sehr geringen Probandenzahlen die wichtigsten *Usability*-Probleme beobachtbar werden. Tests müssen in ein konkretes Anwendungsszenario integriert werden, um explorativen und zielloses Umherklicken zu vermeiden. Als typische Szenarien für den Presenter wurde die selbständige Erstinstallation und Inbetriebnahme der Softwarekomponenten sowie das Navigieren durch eine vorgefertigte Präsentation gewählt. In den abschließenden Tests konnten Probleme festgestellt werden, die vom Entwicklerteam nicht vorhergesehen werden konnten: Handlungsabläufe für die Inbetriebsetzung waren den Testnutzern teils nicht eindeutig verständlich, Informationen wurden nicht oder nur spät identifiziert. Die Probleme wurden behoben und die Applikation anschließend auf dem Market veröffentlicht.

5 Fazit

Das hier beschriebene Entwicklungsverfahren bedient sich bei bestehenden Modellen (*User-Centered Design*, *Design Thinking*) und passt diese an die Anforderungen der mobilen App-Entwicklung, im speziellen dem noch jungen Distributionsweg der App-Märkte an. Die

Vorgehensweise ähnelt dabei den Design-Methoden, wie sie auch bei erfolgreichen Unternehmen wie Google zum Einsatz kommen (Au et al. 2008). Dabei ist wichtig, dass die Entwicklung von Apps schon in der Ideenfindungsphase scheitern kann, wenn ohne Bezug zu den Nutzern entwickelt wird. Ebenso entscheidend ist eine sorgfältige Testphase vor der Veröffentlichung, da anfänglich schlechte Bewertungen eine Applikation von Anfang an zum Scheitern verurteilen können. Nach einem nur insgesamt zweiwöchigen Entwicklungsprozess mit vier Teammitgliedern, die nicht in Vollzeit an der App arbeiteten, wurde die Applikation in einer kostenlosen Version bereits über 1200 mal heruntergeladen, die kostenpflichtige Version ohne Laufzeitbeschränkung konnte in den ersten zwei Wochen über 50 Verkäufe erzielen. Die Website zur Anwendung „Presenter“ ist verfügbar unter der Adresse *http://www.small-worlds.de/presenter*.

Die Entwicklung wurde durch folgende Maßnahmen effizient gestaltet:

- Prototyp geht vor Dokumentation – Papierberge wurden vermieden, das Verständnis der Benutzeroberfläche wurde in Workshops bestimmt und auf Papier herausgearbeitet - eine gewisse Analogie zu den agilen Methoden in der Softwareentwicklung (Srinivasan & Lundqvist, 2009) ist nicht von der Hand zu weisen.
- Effizientere Kommunikation und Entwicklung durch frühe Klärung der funktionalen Anforderungen in der Designphase – Das Team profitiert von einer frühen Verständigung auf gemeinsame Ziele
- Interface Guidelines erleichtern Designentscheidungen – Diskussionen wurden vermieden, wenn klare Empfehlungen seitens des Plattformherstellers vorhanden waren
- Design Patterns auf Codeebene erleichterten Anpassungen, Test und Wartbarkeit des Codes
- Laufende informelle Usability-Tests kosten wenig Zeit und liefern gute Erkenntnisse.

Literaturverzeichnis

Accenture Technology Labs. (2011). Accenture Technology Vision 2011: http://nstore.accenture.com/technologyvision/data/pdfs/TechVision2011_Report_v6_090211_lores.pdf. - Letzter Aufruf am 28. März 2011.

Alben, L. (1996). Quality of experience: defining the criteria for effective interaction design. In: ACM Interactions, 3/1996, 11-15.

Apple Inc. (2011a). iOS Dev Center - Apple Developer: http://developer.apple.com/devcenter/ios/index.action. - Letzter Aufruf am 02. April 2011.

Apple Inc. (2011b). iOS Human Interface Guidelines: http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html. - Letzter Aufruf am 28. März 2011.

Au, I., Boardman, R., Jeffries, R., Larvie, P., Pavese, A., Riegelsberger, J., et al. (2008). User experience at google. Proceeding of the twenty-sixth annual CHI conference extended abstracts on Human factors in computing systems - CHI ’08 (p. 3681). New York, New York, USA: ACM Press.

Burnette, E. (2010). Hello, Android: Introducing Google's Mobile Devel opment Platform (Pragmatic Programmers). Pragmatic Bookshelf.

Craft, B., & Cairns, P. (2009). Sketching sketching: outlines of a collaborative design method. Proceedings of the 23rd British HCI Group Annual Conference on People and Computers: Celebrating People and Technology (p. 65–72). Swinton: British Computer Society.

Dahm, M. (2005). Grundlagen der Mensch-Computer-Interaktion. Pearson Studium.

Ginsburg, S. (2010). Designing the {iPhone} User Experience: A {User-Centered} Approach to Sketching and Prototyping {iPhone} Apps. Addison Wesley.

Google. (2011). User Interface Guidelines: http://developer.android.com/guide/practices/ui_guidelines/index.html. - Letzter Aufruf am 29. März 2011.

Gray, D., Brown, S., & Macanufo, J. (2010). Gamestorming: A Playbook for Innovators, Rulebreakers, and Changemakers. Sebastopol/CA: O'Reilly.

Klauser, K., & Walker, V. (2007). It ' s About Time : An Affective and Desirable Alarm Clock. Design, (August), 22-25.

Krug, S. (2009). Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability Problems (p. 168). New Riders Press.

Matos, V., & Grasser, R. (2010). Building applications for the Android OS mobile platform: a primer and course materials. J. Comput. Small Coll., 26(1), 23–29. USA: Consortium for Computing Sciences in Colleges.

Meinel, H., Weinberg, C., & Plattner, U. (2009). Design Thinking. Hasso Plattner Institut.

Norman, D. A. (1988). The Psychology Of Everyday Things (p. 272). Basic Books.

Research in Motion Limited. (2010). BlackBerry Smartphones UI Guidelines: http://docs.blackberry.com/en/developers/deliverables/17965/Constraints_of_designing_for_mobile_devices_1017065_11.jsp. - Letzter Aufruf am 26. März 2011.

Royce, W. W. (1987). Managing the development of large software systems: concepts and techniques. Proceedings of the 9th international conference on Software Engineering (p. 328–338). Los Alamitos: IEEE Computer Society Press.

Selby, R. W. (2007). Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research (Practitioners). Wiley-IEEE Computer Society Pr.

Shneiderman, B., & Plaisant, C. (2009). Designing the User Interface: Strategies for Effective Human-Computer Interaction (p. 624). Pearson.

Snyder, C. (2003). Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces (Interactive Technologies) (p. 408). Morgan Kaufmann.

Srinivasan, J., & Lundqvist, K. (2009). Using Agile Methods in Software Product Development: A Case Study. 2009 Sixth International Conference on Information Technology: New Generations (pp. 1415-1420). Washington: IEEE.

Warfel, T. Z. (2009). Prototyping: A Practitioner's Guide . Rosenfeld Media.

Wikipedia (2011). Android Market: http://de.wikipedia.org/wiki/Android_Market. - Letzter Aufruf am 03. April 2011.