Mobile App Testing – Nutzerfeedback automatisiert miteinbeziehen

Frank Elberzhager, Konstantin Holl, Simon André Scherr
Fraunhofer IESE, Fraunhofer-Platz 1, 67663 Kaiserslautern
{frank.elberzhager, konstantin.holl, simon.scherr}@iese.fraunhofer.de

Zusammenfassung:
Um eine innovative mobile App schnell auf den Markt zu bringen, nehmen Anbieter in Kauf, sie mit eingeschränkter Qualität bzw. Funktionalität als sogenanntes Minimum Viable Product (MVP) zu entwickeln. Die Gratwanderung besteht darin, sich trotz Zeit- und Kostendruck auf die richtigen Features und Qualitäten zu konzentrieren. Andernfalls werden mobile Apps trotz innovativer Idee auf dem Markt nicht akzeptiert. Um den MVP-Ansatz zu optimieren, sollen Nutzerrückmeldungen früh und schnell berücksichtigt sowie auf typische Mängelaspekte fokussiert werden. In diesem Artikel soll ein Framework vorgestellt werden, welches textuelle Nutzerrückmeldungen sowie das Nutzerverhalten bei Verwendung der App erfasst und dieses auswertet. Die automatisch ausgewerteten Rückmeldungen liefern Informationen hinsichtlich der Verwendung und des Zustands der App sowie explizite Probleme und Featurewünsche der Nutzer. In der Weiterentwicklung der mobilen Apps stellen die Rückmeldungen entscheidende Hinweise für Personen in der Qualitätssicherung und der Entwicklung dar. Dies führt zu höherer Qualität und somit frühzeitig zur Optimierung mobiler Apps

Schlüsselworte: Qualitätssicherung, Mobile Apps, Nutzerrückmeldungen, Minimum Viable Product

1. Einleitung und Motivation

Mobile Applikationen gewinnen in der täglichen Anwendung immer mehr an Bedeutung. 353 Millionen Laptops und Desktop Computer stehen 563 Millionen verkauften Smartphones und Tablets gegenüber [1]. Die zunehmende Verbreitung von Smartphones und Tablets ermöglicht es unter anderem, bestehende Geschäftsprozesse durch innovative Funktionen mobiler Applikationen (wie z.B. standortabhängige Eingabevorschläge) anzureichern, um die Durchführung des Geschäftsprozesses zu unterstützen. Dabei ist ein fehlerfreier Ablauf zu gewährleisten. Zu den Top 10 der häufigsten Absturzursachen mobiler Applikationen gehört u.a. der Übergang aus einem Netz in ein anderes [2]. Ein Verbindungsabbruch in einem kritischen Moment, z.B. während einer mobilen Produktionssteuerung [3], kann zu einer Prozessunterbrechung und somit zu erheblichen Kosten führen. Daher wird ein Qualitätssicherungsansatz benötigt, der explizit mobil-spezifische Probleme findet und den Besonderheiten mobiler Entwicklungsprozesse gerecht wird. Typisch für den mobilen Bereich sind kurze Time-to-Market Zeiten und geringe Budgets für Entwicklung und insbesondere für Qualitätssicherung.

Zur erfolgreichen Etablierung innovativer Produkte und Dienstleistungen am Markt ist der richtige Zeitpunkt essentiell. Werden Trends verschlafen oder Innovationen zu spät marktreif, besteht das Risiko, dass getätigte Investitionen nie einen Return on Investment bringen. Um diesem Problem zu begegnen, verfolgen immer mehr Firmen die Ideen des Lean Startup [4]. Anstatt vollkommen ausgereifte Produkte auf den Markt zu bringen, deren Entwicklung enorme Zeit und Kosten verlangen können, ist hierbei das Ziel möglichst schnell und kostengünstig ein initiales, überlebensfähiges Produkt am Markt zu platzieren – durch ein Minimum Viable Product [4] (kurz: MVP) – und auf Basis von Erfahrung und Kundenrückmeldungen dieses sukzessive im weiteren Softwareentwicklungsprozess zu erweitern und zu optimieren.

2. Hintergrund

Bei der Entwicklung eines MVP nimmt der Anbieter bewusst die fehlende Reife des Produkts in Kauf, um sich im Gegenzug schnell mit einer innovativen Idee am Markt zu platzieren.

Die Herausforderung ist hierbei, dass das Produkt zur Akzeptanz schnell durch einen geeigneten Ansatz zur Qualitätssicherung optimiert werden muss. Im Umfeld von Business-to-Consumer-Produkten sind die Erfahrungen und Meinungen – welche jedoch häufig nicht schnell genug erfasst werden – von großen Nutzergruppen relevant. Dies betrifft insbesondere Dienstleistungen, die über mobile Applikationen realisiert werden, da hier schnell Nutzerzahlen im Zehntausenderbereich erzielt werden. Entsprechend können potenzielle Mängel, wie fehlerhaft implementierte, fehlende oder unvollständige Features, zu erheblichen Schäden für unterschiedliche Stakeholder (z.B. Anbieter, Nutzer) führen.

Daher wird eine fokussierte Qualitätssicherung benötigt, die hinsichtlich der Besonderheiten der MVP-basierten Entwicklung mobiler Applikationen (z.B.

Budgetrestriktionen und kurze Entwicklungszeiten) besser unterstützt. Durch einen fokussierten Qualitätssicherungsansatz als Teil einer MVP-basierten Entwicklung, wird neben der höheren Qualität der zu entwickelnden mobilen Applikationen ebenfalls eine kürzere Time-to-Market erwartet. Die frühe Erfassung und Verarbeitung von Nutzerrückmeldungen sichert die Marktakzeptanz und führt zu einer größeren Verbreitung und zu einer höheren Nutzung der mobilen Applikation.

3. Nutzerrückmeldung

Zunächst muss geklärt werden, welche Arten von Rückmeldungen überhaupt existieren, und wie diese eingeordnet werden können. Erst dann ist es möglich, unterschiedliche Rückmeldungen adäquat (z.T. automatisiert) zu erfassen, auszuwerten, und einen Mehrwert für die weitere Entwicklung und die Qualität zu ziehen. Im Folgenden unterscheiden wir vier Kriterien für die Einordnung von Rückmeldungen.

3.1 Wahrnehmung

Bei Wahrnehmung unterscheiden wir zwischen explizit und implizit. Explizite Wahrnehmung bedeutet, ein Nutzer erstellt bewusst Feedback, z.B. durch das Verfassen von textuellen Informationen in einem App-Store. Implizite Wahrnehmung meint dagegen, dass Rückmeldungen unbewusst generiert werden, beispielsweise durch hintergründiges protokollieren von Nutzungsdaten. Der Nutzer kann in diesem Fall zwar einmal bewusst die Entscheidung getroffen haben, dass hintergründig Daten gesammelt werden, nimmt dies bei der Nutzung von Apps aber nicht ständig wahr.

3.2 Modus

Beim Modus unterscheiden wir zwischen direkt und indirekt. Direkt meint dabei Rückmeldungen, welche unmittelbar an die App-Entwickler gerichtet sind. Das kann zum Beispiel das Dokumentieren eines Eintrags in einem Forum sein, welches die App Entwickler online bereitstellen. Dagegen bedeutet indirekt, dass Feedback nicht unmittelbar an die App-Entwickler gerichtet ist, also zum Beispiel das Verfassen eines Blogeintrags, welcher in einem unabhängigen Online-Portal veröffentlicht wird. Die App-Entwickler können dieses auch lesen, allerdings ist bei diesem Modus das Feedback nicht direkt an sie gerichtet.

3.3 Datentyp

Die Rückmeldungen können unterschiedlicher Art sein. Wir unterscheiden zwischen Inhaltswert und Skalenwert. Ersteres bedeutet Feedback, welches nicht durch eine Ordinalskala abgegeben wird, beispielsweise das Festhalten eines Eintrags in einem Forum mittels Text und Bild. Skalenwert bedeutet hingegen Feedback, welches sich durch eine Ordinalskala ausdrücken lässt. Hier sind z.B. „Sterne“-Bewertungen zu nennen, wobei 1 Stern sehr schlecht und 5 Sterne sehr gut bedeuten kann.

3.4 Zweck

Das letzte Kriterium bei der Einordnung von Feedback ist der Zweck. Zunächst gibt es pauschales Feedback. Das bedeutet, es handelt sich um ausschließlich beurteilende Rückmeldungen, z.B. in Form einer Aussage wie „die App ist schlecht“. Daneben kann es spezifisches Feedback geben, welches konkrete Vorschläge zur Verbesserung, Begründungen für eine Bewertung bzw. Erklärungen für Fehlverhalten beinhaltet. Beispiele sind hier Aussagen wie „die App ist schwer zu bedienen, weil die Buttons zu klein sind“.

3.5 Beispiele für konkrete Rückmeldungen

Die vier genannten Kriterien lassen sich in einer Matrix aufspannen, in die konkrete Feedbackarten einsortiert werden können. Damit zeigt sich folgendes:

- Für welche Kombinationen ist Feedback überhaupt zu erhalten (d.h., nicht alle 4-Tupel sind überhaupt möglich)
- Die Möglichkeit des gezielten Konzentrierens auf das Feedback, welches am sinnvollsten zu bekommen ist
- Die Basis für weitere Fragestellungen wie „wie gelange ich an das entsprechende Feedback?“, „wie werte ich das Feedback aus?“, oder „was für Erkenntnisse bekomme ich aus dem Feedback heraus?“

Unter der Prämisse, dass Feedback für die eigenentwickelte App gewonnen werden möchte, stellt Tabelle 1 Beispiele für verschiedene Arten von Feedback dar. Bei dem 4-Tupel ‚explizit-direkt-inhaltswert-pauschal‘ ist zum Beispiel In-App Feedback, Feedback über eine Hotline oder auch Feedback via Email denkbar. Neben einer Vielzahl von Möglichkeiten explizit Feedback zu geben, ist eine weitere wesentliche Informationsquelle implizit erfasstes Feedback mittels einer Komponente auf dem mobilen Gerät bzw. direkt implementiert in der App. Diese sammelt Nutzungsdaten und macht sie dem App-Entwickler verfügbar. Diese Komponente wird hier Agent genannt.

Verschiedene Arten von Feedback lassen sich zudem mehreren Bereichen zuordnen, beispielsweise kann Feedback in einem Store pauschal oder auch spezifisch sein. Zuletzt sind verschiedene Kombinationen nicht möglich; so kann eine Bewertung der App (Zweck)

	Wahrnehmung	Explizit	Explizit	Implizit	Implizit
	Modus	Direkt	Indirekt	Direkt	Indirekt
Datentyp	**Zweck**				
Inhaltswert	Pauschal	• In-App • Hotline • Email …	• Store • Forum • Blog …	• Agent • Nutzerbeobachtung	n/a
Inhaltswert	Spezifisch	• In-App • Forum • Portal …	• Store • Forum • Blog …	• Agent • Nutzerbeobachtung	n/a
Skalenwert	Pauschal	• In-App • Portal • Hotline …	• Store • Magazin • Studie …	• Agent	n/a
Skalenwert	Spezifisch	n/a	n/a	n/a	n/a

Tabelle 1: Rückmeldungsarten

mittels einer Skala (Skalenwert) nichts Spezifisches aussagen (s. untere Zeile in Tabelle 1).

4. Ablauf bei der Erfassung und Auswertung von Feedback

Die grundsätzliche Idee des Opti4Apps Frameworks ist es, unterschiedliche Rückmeldungen einer als MVP entwickelten App zu gewinnen, dieses (semi-)automatisch auszuwerten und aus den gewonnenen Information frühzeitig einen Mehrwert für die Qualität und die weitere Entwicklung zu ziehen.

Zur Erfassung von Rückmeldungen wird das Opti4Apps Framework wie folgt genutzt: Der Agent sammelt Daten zum Usage Mining. Der App Benutzer kann zusätzlich Informationen zur Verbesserung der App geben, entweder direkt an den Entwickler gerichtet wie beispielsweise über ein Kontaktformular, eine gezielte Feedbackfunktion, Bewertungen in App Stores, oder einfach ein Ausdruck der persönlichen Meinung wie Postings in sozialen Medien oder Webforen.

Die dadurch entstehenden Rohdaten werden mittels Data Mining analysiert. Daraus ergeben sich zum einen Verbesserungsindizien für die App, zum anderen werden die Daten genutzt, um die Interpretationsregeln für die Daten sowie die bisher verzeichneten Fehlermuster zu ergänzen und zu überarbeiten. Aus den Verbesserungsindizien werden wiederum von einer verwaltenden und entscheidenden Rolle, beispielsweise einem Testmanager, nach Prüfung neue Tickets erstellt, die auf dem dokumentierten Fehlverhalten sowie neuen Anforderungen basieren, die dann an die Entwicklung weitergegeben werden. Die Tester der App wiederum berücksichtigen die gewonnenen Erkenntnisse beim Erstellen und Ausführen von Testfällen bzw. die Inspektoren bei ihren Inspektionen.

5. Architektur

Das Opti4Apps Framework lässt sich strukturell in drei Ebenen aufteilen: Mobilgerät, Backend, und Frontend (siehe Abbildung 1). In der jeweiligen App auf dem Mobilgerät werden aktiv Daten für das Usage Mining gesammelt. Hierbei gesammelte Daten werden an das Backend weitergegeben. Dort findet das eigentliche Data Mining auf den Rohdaten statt. Zusätzlich zum Usage Mining wird im Backend Text Mining auf mittels eines Crawlers gesammelten Daten betrieben. Hierbei werden textuelle Informationen rund um die Nutzung der App maschinell nach Hinweisen bzgl. Qualitätsverbesserungen und -problemen analysiert. Ergebnisse beider Analysen lassen sich dann im Front-End in Form eines Dashboards darstellen und sind so in aggregierter Form für die an der Qualitätssicherung bzw. Entwicklung beteiligten Personen einsehbar.

5.1 Mobilgerät

Auf dem Mobilgerät ist die App, die bewertet werden soll. Die Nutzung der App wird von einem Agenten überwacht. Dieser Agent enthält sowohl einen Protokollierer, der Ereignisse innerhalb der App systematisch loggt, als auch den Pre-Usage Miner, der eine Vorauswahl an Daten trifft, die für das spätere Text Mining auf dem Backend relevant sind.

Daten, die der Protokollierer festhält, sind sowohl Gerätedaten wie Sensordaten, Leistungsdaten oder Gerätezustände, als auch Interaktionsdaten wie beispiels-

weise eine Navigationsfolge innerhalb der App, Nutzerinteraktionen mit Elementen auf dem Display oder aktuell angezeigte Elemente.

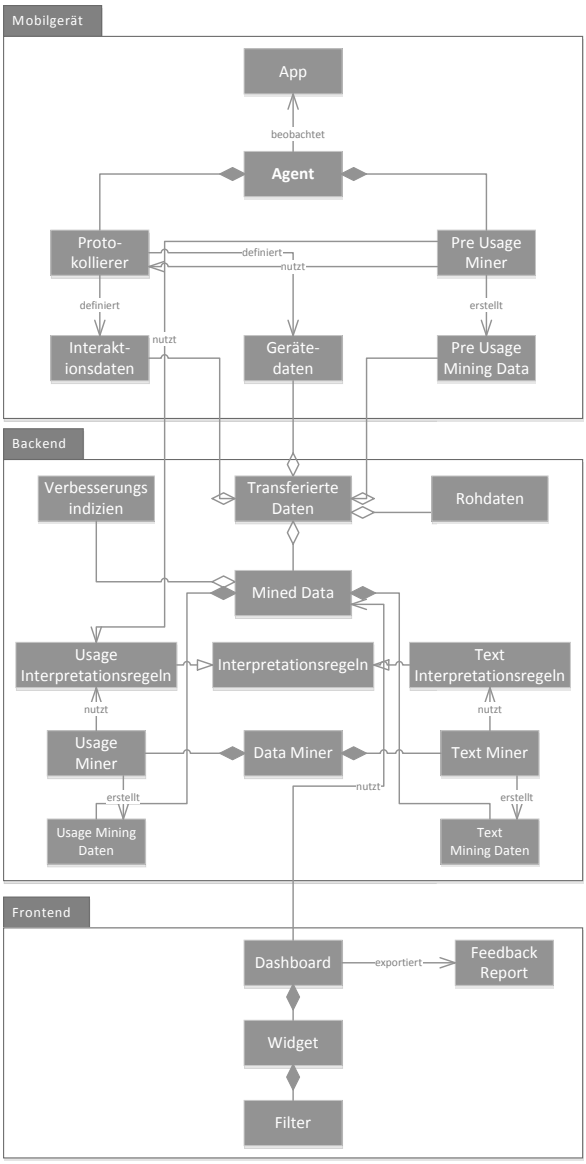

Abbildung 1: Opti4Apps Komponenten

5.2 Backend

Die auf dem Mobilgerät gesammelten Daten werden schließlich an das Backend übertragen und vom Usage Miner mit Techniken des Data Minings weiterverarbeitet.

Nutzer von Mobilgeräten geben textuelles Feedback an zahlreichen Orten und in vielerlei Formen ab. Dieses Feedback erfasst und analysiert die Text Miner Komponente systematisch, um so Empfehlungen zu Verbesserungen der Anwendungsqualität zu geben.

Beiden Mining Komponenten ist gemein, dass sie für ihre Auswertungen auf Interpretationsregeln zum Verarbeiten der Daten zugreifen.

Durch die Kombination aus den Ergebnissen des Usage und Text Mining entsteht ein umfassenderes Bild, welche Maßnahmen zur Verbesserung der Qualität bzw. welche neuen Features gewünscht werden.

5.3 Frontend

Die Ergebnisse des Data Minings werden in einem Dashboard-basierten Frontend zugänglich gemacht. Das Dashboard enthält zahlreiche Widgets, die einzelne Informationsstränge anzeigen. Innerhalb dieser Widgets können optional Filter angewendet werden, um nur bestimmte Arten von Daten anzuzeigen oder eine zeitliche Einschränkung vorzunehmen.

5.4 Diskussion

Während eine MVP-Entwicklung zwar frühe Rückmeldungen der Nutzer ermöglicht und eine sehr schnelle Time-to-Market mit sich bringt, leidet oftmals die Qualität des Produkts. Bei der klassischen Entwicklung gibt es hingegen keine frühen Rückmeldungen und eine lange Time-to-Market.

Kernaspekt der Qualitätssicherung bei Opti4Apps ist eine Methodik, die auf abgeleiteten Fehlermustern basiert. Diese Fehlermuster definieren die Verbindung von typischen Fehlerursachen mit deren häufigen Fehlerwirkungen und sind für die Qualitätssicherung der entwickelten mobilen Dienstleistungsapplikation im Sinne des Testens sowie für die Qualitätssicherung der Anforderungen im Rahmen von Reviews geeignet.

Durch Opti4Apps werden die Vorteile der MVP-Entwicklung realisiert und erweitert, indem ein Framework auf Basis einer automatisierbaren Methode zur Rückmeldungserfassung und -auswertung entwickelt und eingesetzt wird.

Acknowledgements

Dieser Beitrag wurde erstellt im Kontext des Projekts Opti4Apps, gefördert durch das Bundesministerium für Bildung und Forschung (Förderkennzeichen: 02K14A182).

Referenzen

[1] Statista GmbH, Prognose zum Absatz von Tablets weltweit in den Jahren 2010 bis 2019. http://de.statista.com/statistik/daten/studie/165462/umfrage/prognose-zum-weltweiten-absatz-von-media-tablets, 2015

[2] Aguilera, Ray, Top 10 Reasons iOS and Android Apps Crash. Mashable. http://mashable.com/2012/07/20/why-ios-android-apps-crash, 2012

[3] Zühlke, Karin, Mit der "Android-App" von unterwegs die Produktion sicher überwachen. WEKA FACHMEDIEN GmbH. http://www.elektroniknet.de/elektronik fertigung/sonstiges/artikel/86749/0, 2012

[4] The Lean Startup, The Movement That Is Transforming How New Products Are Built And Launched, http://theleanstartup.com, 2014