MOBILE INTERAKTION: MUSTER ZUR LÖSUNG ORGANISATORISCHER PROBLEME IN DER APP-ENTWICKLUNG

Mobile Applikationen (Apps) leben von ihrem interaktiven Charakter. Gute Benutzungsoberflächen zu entwickeln, die das Potenzial heutiger Smartphones heben, ist eine schwierige Aufgabe. In diesem Artikel werden einige typische Probleme identifiziert, die Unternehmen häufig davon abhalten, überzeugende mobile Applikationen zu liefern, und es werden Lösungsansätze vorgeschlagen.

der autor

Jörg Pechau
(E-Mail: jop@icnh.de)
ist geschäftsführender Gesellschafter bei der I.C.N.H GbR. Seine Interessen liegen in der Beratung, Entwicklung und im Training rund um die Themen mobile Anwendungen und agile Softwareentwicklung. Daneben forscht und lehrt er über agiles und phasenorientiertes Software-Projektmanagement.

Einfache und gute Benutzungsoberflächen (*User Interfaces – UI*) umzusetzen, ist schwierig und benötigt seine Zeit. Schlechte Benutzungsoberflächen, zu entwickeln ist einfach, wenngleich auch nicht unbedingt schneller. Das sind Binsenweisheiten? Wenn ja, warum finden sich in den verschiedenen Stores für „mobile Applikationen" (App-Stores) so viele Apps, deren Benutzung fast schon schmerzt?

Als Berater sind wir in der angenehmen Situation, Blicke hinter die Kulissen verschiedener Unternehmen werfen zu können. So sehen wir, dass sich in den Unternehmen in der Regel sehr motivierte und gut ausgebildete Menschen ihre Köpfe zerbrechen, wie sie ihre mobile Anwendung am besten entwickeln. Dennoch bleiben die Ergebnisse häufig hinter ihrem Potenzial zurück – und das ist für alle Beteiligten verständlicherweise enttäuschend.

Enttäuschung ist eine Sache, deren Konsequenzen schwer messbar sind. Doch ein Image-Schaden – bis hin zu handfesten wirtschaftlichen Schäden – ist eine andere Sache. Wenn wir den Analysten (vgl. [Loc]) glauben schenken wollen, dann werden 26 % aller Apps, die im Apple AppStore von Kunden geladen wurden, nur einmal gestartet.

Machen Sie sich doch einmal die Mühe und lesen Sie sich einige Kommentare einer nicht so gut bewerteten App auf der mobilen Plattform Ihrer Wahl durch: „Keine Ahnung, wozu diese App gut sein soll" oder „Stürzt nach dem letzten Update andauernd ab". Das sind noch eher freundlichere Kritiken. Ob diese im Einzelnen gerechtfertigt sind oder nicht, sei dahin gestellt. Es bleibt: Die Kunden mobiler Apps sind wenig verzeihend. Der erste Eindruck einer App zählt und bleibt hängen. Das kann das Image schädigen. Das kann aber auch bedeuten, dass ein Unternehmen deutlich weniger Umsatz mit seiner App macht, als geplant.

Die Situation stellt sich ähnlich dar, wenn die Kunden aus dem eigenen Unternehmen kommen, z. B. für eine vertriebsunterstützende App. Die hausinternen Kunden wandern zwar nicht ab, weil sie ja gezwungen sind, die App zu verwenden. Doch eigentlich wollen wir Kollegen erleben, die statt mäßig verborgener Unzufriedenheit lieber ihre neue App begeistert vorführen und damit für ihr Unternehmen werben. Auch interne Kunden sind Kunden und auch hier kann Image- oder wirtschaftlicher Schaden entstehen – und sei es „nur" ein verschenktes Investment.

Die Arbeit erlaubt es meinen Kollegen und mir, Vergleiche zwischen verschiedenen Unternehmen zu ziehen. Da wir meist nur für kurze Zeit Teil der Organisation sind, haben wir den Luxus einer externen Perspektive. Vor diesem Hintergrund konnten wir beobachten, dass sich bestimmte Probleme wiederholen und dass sich Muster erkennen lassen. Dabei fiel uns auf, dass sich die Ursachen der beobachteten Probleme in der Mehrzahl der Fälle auf der Organisations- statt auf der Kompetenzebene finden lassen. Ausnahmen bestätigen bekanntlich – und auch hier – die Regel. Da für die pathologischen Fälle „mangelnde Motivation" oder „Inkompetenz" die Lösungen auf der Hand liegen, kümmern wir uns stattdessen lieber um die interessantere Frage: Was sind typische organisatorische Probleme und wie können wir sie vermeiden?

Drei Muster

Von den typischen organisatorischen Problemen werde ich mich hier auf die drei am häufigsten anzutreffenden Vertreter beschränken.[1])

1. Never walk alone ...

„Am Ende geben wir es dem Designer, der macht das dann hübsch. Fangt einfach schon mal an."

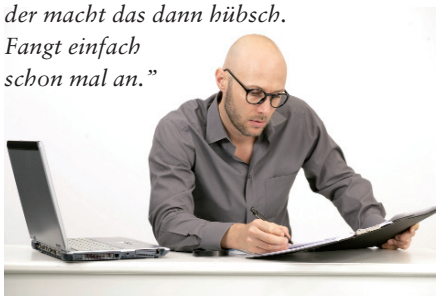

Abb. 1: Architekt mit Plänen (Foto: Peter Atkins, www.fotolia.de).

Kontext: Es kommt immer wieder vor, dass ein im Interaktions- und Grafikdesign unerfahrenes Softwareentwicklungsteam die App ohne entsprechende Begleitung realisieren soll, weil entsprechende Kompetenz nicht im ausreichenden Maße zur Verfügung steht.

Problem: Die App-Entwicklung darf nicht ohne Interaktions- und Grafikdesign-Unterstützung starten.

In den meisten Unternehmen, sind Grafikdesigner knapp. Speziell ausgebildete Interaktionsdesigner – eine vergleichsweise junge Disziplin – sind noch seltener zu fin-

[1]) *Für die Beschreibung orientieren wir uns an dem „Organizational-Muster"-Format, wie es in[Cop04] verwendet wird, d. h. Titel/Bild/Zitat, Kontext, Problem, Lösung, Diskussion.*

den. Die Designer werden dementsprechend für alles mit einem Bezug zu Design in die Pflicht genommen: Das reicht von der Ausgestaltung des Corporate Designs, bis hin zum Entwurf der Firmen-Webseiten oder von Marketingmaterial. Sie sind nicht Teil eines, sondern einer ganzen Anzahl von Projekten. Das hat typischerweise zur Folge, dass sie nicht kontinuierlich bei einer App-Entwicklung mitarbeiten können. Stattdessen liefern sie – wenn überhaupt – einen statischen Entwurf zum Entwicklungsbeginn und am Ende versuchen sie, das Ergebnis zu veredeln.

App-Entwicklungen sind hochgradig interaktive Anwendungen, die vor allem durch ihre Optik und Interaktion durch den Nutzer erlebt werden. Apps zu entwickeln heißt, permanent die Rückkopplung mit dem Anwender zu suchen und sein Design immer wieder anzupassen (siehe unten, „Probieren geht über studieren"). Nicht alles, was in Photoshop gut aussieht, fühlt sich in der Realität auch gut an (siehe unten, „Apps sind physische Anwendungen"). Die Entwicklung einer App wird gerade dann an Dynamik gewinnen, wenn sie erfolgreich in einen Marktplatz oder App-Store eingestellt wurde und eifrig genutzt wird. Das heißt, solange die App lebt, muss eine Kombination aus Design- und Entwicklungskompetenz zur Verfügung stehen.

Dem Team diese Designkompetenz vorzuenthalten bzw. es nur punktuell zu unterstützen, gefährdet den Projekterfolg und heißt, die Aufgabe des Designers nicht ernst zu nehmen oder zu verstehen. Es besteht die Gefahr, dass das Entwicklungsteam deutlich mehr Aufwand leisten muss, um seine fehlende Kompetenz zu kompensieren bzw. zu erlernen, sofern es dazu aus eigener Kraft überhaupt in der Lage ist. Das, was den Erfolg einer App ausmacht, wird damit dem Zufall überlassen.

Lösung: Das Entwicklungsteam muss bereits über Kompetenz für Interaktions- und Grafikdesign verfügen.

Wie kann man das erreichen? In einer idealen Welt gehört dem Team ein entsprechender Designspezialist an, dessen Hauptqualifikation und -aufgabe das Design der App ist. Ergänzend sollte bei der Zusammenstellung des Teams zumindest ein Teil der Entwickler bereits eine Affinität zur UI-Entwicklung aufweisen, um entsprechende Designaufgaben lösen zu können. Vor allem aber können entsprechend qualifizierte Entwickler als Sparring-Partner für den oder die Designspezialisten dienen, da sie die Machbarkeit von Designentwürfen aus einer technischen Perspektive beurteilen können.

So wie der Entwickler ein bisschen Designer sein muss, muss der Designer auch ein wenig Entwickler sein. Das heißt, beide wissen, was machbar ist und was nicht. Und man versteht sich auch besser. Die Welt ist selten ideal. Deswegen bietet es sich als Alternative an, entsprechende interessierte Softwareentwickler neben der Entwicklung auf der mobilen Plattform auch im UI-Design für diese Plattform auszubilden und sie entsprechend zu coachen. Diese Anforderungen gelten natürlich auch für Teams, die extern „angeheuert" werden.

Diskussion: Als Ergebnis erhalten wir ein autonomes Team, das seine App dauerhaft betreuen kann und das dazu in der Lage ist, ein überzeugendes *Look&Feel* abzuliefern. Hausinterne Designer müssen sich nicht zwischen ihren Projekte zerreißen und finden in den Entwicklungsteams kompetente Ansprechpartner.

Dabei sollte man allerdings darauf achten, dass sich in einem autonomen Entwicklungsteam das Design nicht verselbstständigt und vom Corporate- oder Marken-Design des Unternehmens abkoppelt. Das heißt, zwischen Entwicklungsteam und dem für die CI zuständigen Bereich muss auch bei autonomen Teams ein regelmäßiger Dialog stattfinden. Wird die Entwicklung von externen Teams übernommen, ist es eine Herausforderung an die Vertragsgestaltung, die oben beschriebene Anforderungen an die Teamzusammenstellung und die Kontinuität in einem Vertrag abzubilden. Wenn externe Spezialisten für die Designaufgaben das Team verstärken, ist das Coaching des Teams in UI-Designfragen eminent wichtig. Anderenfalls wird sich am Grad der Autonomie des Teams nichts ändern.

2. Apps sind physische Anwendungen ...

„Meine Finger sind nicht klein genug für diese Knöpfe!"

Abb. 2: *Hände (Foto: Ralf-Udo Thiele, www.fotolia.de).*

Kontext: Die Anwender erleben eine mobile App und ihr Smartphone oder Tablet als eine Einheit. Doch das UI-Design wird in der Form von statischen Bildschirmabfolgen entworfen, in denen die Endgeräte nur den die Größe bestimmenden Rahmen für das UI bilden.

Problem: Mobile Anwendungen dürfen nicht ausschließlich am „Reißbrett" entwickelt werden.

Das Reißbrett (z. B. Photoshop) bildet einen guten Startpunkt. Wenn sich das UI-Design allerdings darin erschöpft, besteht die Gefahr, dass mit Smartphones mögliche Interaktionsformen gar nicht oder falsch genutzt werden.

Eine Multitouch-Geste zusammen mit unterstützender Animation kann viele klassische Klicks vermeiden helfen und die Bedienung der App zu etwas machen, was Spaß bringt. Wir können die Lage des Geräts im Raum und die Beschleunigung des Geräts feststellen und dies in unsere Interaktionen mit einbeziehen sowie dem Nutzer akustisch oder durch Bewegungsimpulse Feedback geben, z. B. wenn wir ein Peripherie-Gerät fernsteuern. Uns stehen integrierte Geräte wie Kompass oder Kameras, gerüchtehalber sogar ein Telefon, zur Verfügung, die wir in die Bedienung der Anwendung integrieren können.

Entsprechende Interaktionen zu definieren, setzt voraus, diese zu kennen. Sind sie bekannt, sind sie auf dem „Reißbrett" in ein UI-Design durchaus anotierbar, doch die gewählten Interaktionen bleiben in dieser statischen Form schwer testbar. Daher ist es wichtig, frühzeitig Prototypen zu bauen.

Auf der anderen Seite soll die App den Erwartungen an Apps der Zielplattform entsprechen: Eine Android-App sollte sich wie eine Android-App und eine iPhone-App eben wie eine typische iPhone-App anfühlen. Einfach nur kleine Webseiten zu zeigen, weil das Design nur Webseiten produzieren kann, ist ein fauler Kompromiss. Des Weiteren muss das UI der Aufgabe angemessen sein (in DIN EN ISO 9241-110). Offensichtlich hängt die Aufgabenangemessenheit einer Benutzungsoberfläche dabei von der zu erledigenden Aufgabe ab. Hier ein Beispiel: Will ich eine Uhrzeit eingeben, dann liefern die bekannten Zahlenräder (*Spinner*) eine gute User-Experience. Muss ich sehr viele Uhrzeiten nacheinander eingeben, geht das in diesem Fall zu Lasten der Benutzbarkeit.

Ein Gefühl für das Zusammenwirken von UI, Interaktionsformen und Gerät und dafür, ob diese Kombination erwartungskonform und den Aufgaben angemessen ist, kann auf dem Papier allein nicht entste-

hen.

Verzichten wir aus Zeitdruck oder aus Mangel an Ressourcen auf Interaktionsdesign und beschränken wir uns darauf, statische UI-Abfolgen zu entwerfen und umzusetzen, wird unsere App ärmer. Verstoßen wir mangels Wissen oder aufgrund falscher Annahmen gegen die auf dieser Plattform üblichen Interaktionsmuster oder wählen wir unübliche Interaktionsmuster aus, wird die App schwerer bedienbar.

Ist eine App im beschriebenen Sinne arm oder schwer bedienbar, wird sie weder uns noch unsere Kunden begeistern.

Lösung: Unsere Anwendung soll interaktiv und intuitiv werden – dafür müssen wir das Reißbrett verlassen, das Design in die Hand nehmen und die richtigen Interaktionsformen erarbeiten.

Am Anfang steht auch hier ein Entwurf auf (digitalem) Papier. Ziel ist es, die Bildschirmabfolgen zu finden und sich erste Gedanken über die Interaktionen und Animationen zu machen. Dabei kann es helfen, die UIs mit Papier und Schere tatsächlich nachzubauen, Elemente zu platzieren, zu modifizieren, entfernen usw. Allzu großer Realismus ist hierfür nicht erforderlich – er kann sogar ablenken, indem wir uns zu früh Fragen des grafischen Designs statt des Interaktionsdesigns stellen. Es reicht aus, wenn wir statt mit dem vollständigen UI-Entwurf mit einen Ausschnitt beginnen, um den Stil unserer App zu finden.

Sobald wir erste belastbare Entwürfe haben, ist es wichtig, diese so rasch wie möglich auf dem Zielgerät umzusetzen. Dies kann beispielsweise ein verlinktes PDF- oder ein HTML-basierter, evtl. mit Javascript angereicherter, Klick-Dummy sein. Auch dies sollte nur ein Zwischenschritt bleiben, denn sobald wie möglich sollten wir das UI der App implementieren, um ein echtes Gefühl für die Anwendung zu bekommen – anfänglich ohne fertig implementierte Funktionen, diese werden später Schritt für Schritt parallel hinzugefügt.

Sobald wir das UI auf dem Endgerät anfassen können, sammeln wir Beobachtungen: Was ist intuitiv verständlich und wo läuft der Anwender in die Irre? Wie nähern sich die Anwender spontan dem UI und welche Touch-Gesten erwarten sie? Welches Feedback und welche Animationen helfen, welche lenken ab oder verwirren gar? Welche neuen Ideen kommen uns beim Testen? Welche unserer Annahmen decken sich mit unseren Beobachtungen?

Wird Funktionalität parallel zur Benutzungsoberfläche entwickelt, sollten wir diese, sobald sie verfügbar ist, integrieren und das Antwortzeitverhalten der App betrachten: Sind die Übergänge fließend, startet sie schnell genug? Wo ist Wartezeit vermeidbar und wo nicht? Wenn wir den Anwender zulange warten lassen, beendet er die App und löscht sie gegebenenfalls. Design und Entwicklung müssen das Design derart optimieren, dass an der Stelle, wo Wartezeit nicht akzeptiert wird, diese durch geschicktes Design, Animation und Programmierung vermieden wird.

Diese Beobachtungen nutzen wir, um unseren Prototypen in vielen kleinen Iterationen zu vervollständigen und schrittweise zur eigentlichen App zu verbessern. Diese Arbeit wird uns auch immer wieder bis an unser Reißbrett und die Entwicklungsbank zurückführen. Das ist normal.

Wenn sich das Interaktionsdesign setzt, ist es an der Zeit, das grafische Design zu integrieren. Auch das kann wieder Auswirkungen auf die Interaktionen haben und auch hier werden wir das grafische Design im Sinne einer guten User-Experience in mehreren Schleifen bearbeiten müssen. Es mag sein, dass ein *Corporate UI-Styleguide* kleine Schaltflächen erfordert, doch wenn diese auf dem Zielgerät nicht bedienbar sind, kann das nicht das letzte Wort sein. Mit ihrem Prototyp finden sie es heraus.

Stellen Sie sicher, dass bei diesem Vorgehen genügend Experten dabei sind, die mit dem Design, aber auch mit der Zielplattform, seinen Möglichkeiten und seinen Konventionen vertraut sind, um im Dialog zwischen Designer und Entwicklung das UI zu schärfen. Wenn Sie die Chance haben, beziehen Sie auch die Anwender frühzeitig mit ein. Nehmen Sie sich genügend Zeit, um ein gutes UI zu finden. Kalkulieren Sie dafür grob geschätzt ca. die Hälfte des Gesamtaufwands für Ihre App-Entwicklung ein.

Diskussion: Mit diesem Vorgehen, erhalten wir ein umsetzbares UI, das sich an der Aufgabe und den Erwartungen des Anwenders orientiert und das mit einem gewissen *Touch* daherkommt. Kennen Sie Ihre Anwender nicht, müssen Sie sich anders behelfen, z. B. durch Einsatz von Personas, Szenarien und User-Storys. Finden Sie zum Testen Teammitglieder, die die Sichtweise einer solchen „Persona" einnehmen können.

Die Gefahr bei dem Vorgehen ist, dass das Team sich im Detail und in Schönheit verliert, bevor es sich dem Anwender erstmals stellt – was erheblich Nacharbeiten nach sich ziehen kann.

3. Probieren geht über studieren ...

Abb. 3: *Baustelle Elbphilharmonie (Foto: Matthias Krüttgen).*

Kontext: Das Entwicklungsteam und/oder dessen Stakeholder (z. B. im Vertrieb, Marketing oder Produktmanagement) möchte erst einen möglichst vollständigen Stand der App umsetzen, ehe es dieses den Anwendern zur Verfügung stellt.

Problem: Interaktionsdesign mobiler Anwendungen bedeutet, dass die richtigen Interaktionen gefunden werden müssen und nicht einfach entworfen werden können.

Die Stärke mobiler Apps im Zusammenspiel mit den Smartphones, auf denen sie laufen, ist ihre Interaktivität. Die App reagiert auf den Nutzer, der Nutzer reagiert auf die App und beide treten in Interaktion. Diese ist hochgradig abhängig von der Aufgabe, für die sie entworfen wird. Eine Fernsteuerung für einen Modellhubschrauber benötigt andere Interaktionsformen als eine *VoIP*-Anwendung (*Voice over IP*) oder ein mobiler Business-Intelligence-Client.

Es gibt eine Vielzahl von Techniken, mit denen wir Anwender und ihre Bedürfnisse erfassen und beschreiben können. Diese Anforderungen umzusetzen, ist gut verstandenes Handwerk. Doch erst im Zusammenspiel der App auf dem physischen Endgerät erkennen wir, wie gut oder schlecht Anwender mit unsere App interagieren, und können darauf eingehen. Erst dann erkennen wir, welche Konzepte funktionieren und welche nicht. Die Chancen, dass ein erster Entwurf bereits die Erwartungshaltung der Anwender trifft, ist gering. Einfach formuliert: Wir müssen ausprobieren, welche Interaktionen im

Anwenderkontext funktionieren und welche nicht – auf dem Papier sieht vieles überzeugend aus. Wir bzw. die Anwender müssen die verschiedenen Alternativen *begreifen*, um das optimale UI zu finden.

Dadurch entsteht ein Dilemma: Auf der einen Seite wollen wir den Anwender für uns gewinnen und müssen herausfinden, wie seine Anforderungen am besten umsetzbar sind. Auf der anderen Seite haben wir Sorge, den Anwender durch einen unfertigen Stand oder durch Ausprobieren möglicher Alternativen zu verschrecken.

Lösung: Agieren Sie dicht am Kunden und arbeiten Sie in kurzen Iterationen, liefern Sie häufig, arbeiten Sie transparent und managen Sie die Erwartungen. Mit anderen Worten: Arbeiten Sie agil.

Betrachten wir die einleitende Liste für die Umsetzung der Lösung, schenkt uns zum Beispiel Scrum (vgl. [Sch01]) die meisten Punkte:

- Autonome, sich selbst organisierende Teams
- Ein *Product Owner*, der die Kunden kennt, betreut und im Projekt vertritt
- Regelmäßige Review-Meetings, in denen *Product Owner* und Kunden Feedback geben
- Kurze Iterationen mit potenziell lieferbare Ergebnissen

Wir sind nicht gezwungen, Konzepte lange zu diskutieren, mit der Chance, nach längerer Zeit im größeren Umfang falsch zu liegen, sondern wir können sie im Kleinen schnell ausprobieren und daraus lernen.

Diskussion: Agile Methoden sind nicht in jedem Unternehmen en vogue und Scrum führt sich auch nicht von selbst ein. Doch kann es uns niemand verbieten zu prüfen, was von den agilen Ansätze bei uns umsetzbar ist. Der Schlüssel zum Erfolg sind die kurz getakteten und regelmäßigen Reviews durch die Anwender – diese lassen sich auch in ein klassisch geprägtes Vorgehen integrieren. Wenn unsere Stakeholder Angst vor einem frühen Feedback der Anwender haben, müssen wir eben versuchen, diese an Stelle der Anwender in die Pflicht zu nehmen. Die Hoffnung ist dabei, dass die Stakeholder ihre Haltung bezüglich früher Einbeziehung der eigentlichen Anwender ändern, sobald sie einmal den Wert und die Geschwindigkeit des frühen Ausprobierens am eigenen Leib erlebt haben.

Das Ausprobieren kann mit der Sorge um steigende Aufwände und explodierende Kosten einhergehen, die sich aus dem Ausprobieren ergeben. Aber auch hier sind die Methoden klassischen oder agilen Managements anwendbar: Wir können auch hier mit *Time-Boxing* und Aufwandsdeckeln arbeiten – das Ergebnis ist dann eben so gut, wie es die Rahmenbedingungen zulassen. Einen gewissen Anteil an Änderungsanforderungen durch Anwender-Feedback müssen wir einkalkulieren. Dies nicht zu tun, legt uns auf den ersten Entwurf fest.

Fazit

Die drei aufgelisteten Muster reißen das Thema nur an, sicherlich lassen sich noch mehr finden. Ich würde mich freuen, wenn ich mit diesem Artikel zu einer Diskussion beitragen konnte. Wesentlich ist für mich der folgende Punkt: Interaktive UIs entstehen vor allem aus der Interaktion zwischen Design, Entwicklung und Anwendern. Nur wenn wir diese durch entsprechende Rahmenbedingungen ermöglichen, kann die kombinierte Design- und Entwicklungskompetenz Früchte tragen. Ein gutes UI erfordert von allen Beteiligten relativ hohe Investitionen von Zeit und Aufmerksamkeit. Nur so wird aus einem zufallsgetriebenen Ansatz ein planbares Vorgehen, dessen erster Eindruck stimmt. ■

Literatur & Links

[Ale77] C. Alexander, A Pattern Language: Towns, Buildings, Construction, Oxford University Press 1977

[Cop04] J. Coplien, N. Harrison, Organizational Patterns of Agile Software Development, Prentice Hall 2004

[Gam94] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns – Elements of Reusable Object-Oriented Software, Addison-Wesley 1994

[Loc] Localytics, Report, siehe:
http://www.localytics.com/blog/post/first-impressionsmatter-26-percent-of-apps-downloaded-used-just-once/

[Sch01] K. Schwaber, M. Beedle, Agile Software Development with Scrum, Prentice Hall 2001