mehr zum thema:
www.designmethodenfinder.de

BEOBACHTEN, FRAGEN, KENNENLERNEN: DER WEG ZUR HERAUSRAGENDEN USER EXPERIENCE BEGINNT BEIM BENUTZER

Um eine tolle User Experience zu liefern, muss man den Benutzer kennen. Doch wer ist dieser Benutzer und wie denkt und arbeitet er? UI-Experten auf der ganzen Welt stellen sich diese Frage und haben verschiedene Techniken entwickelt, um mehr über die Benutzer zu erfahren. Wir stellen Ihnen in diesem Artikel einige Techniken vor, mit denen Sie auch Ihr Produkt verbessern können.

der autor

Roland Gelzer
(E-Mail: roland.gelzer@bbv.ch)
ist seit zehn Jahren in der Softwareentwicklung tätig und interessiert sich vor allem für UI-Aspekte.

Adrian Krummenacher
(E-Mail: adrian.krummenacher@bbv.ch)
arbeitet seit elf Jahren als Softwareingenieur bei bbv Software Services in der Schweiz. Seine Interessensschwerpunkte liegen in den Bereichen User Experience und agile Methoden.

Was haben das iPad und ein moderner Kampfjet gemeinsam? Warum arbeitet man lieber mit Windows 7 oder OSX als mit Windows95, selbst wenn der Durchschnittsbenutzer selten mehr mit seinem Computer macht als anno dazumal. Warum verläuft der Verkauf von Bio-Benzin E10 eher schleppend? Die Antwort ist ganz einfach: Die Firmen haben auf die Bedürfnisse und Interessen der Benutzer geachtet (oder eben nicht, wie im Fall von E10).

Einen weiteren Beweis dafür, dass die Ideen der Benutzer wichtig sind, erbringt das iPhone mit seinem eigenen marktwirtschaftlichen Ökosystem. Und es beweist die Aussage des Verhaltensforschers Donald R. Norman bezüglich guter User Experience und Fehlertoleranz: Wenn etwas gut aussieht und sich gut anfühlt, verzeiht man auch den einen oder anderen Fehler (vgl. [Nor02]).

Will man mehr aus seinem Produkt herausholen als die bloße Funktionalität und eine marketingorientierte Feature-Liste, so müssen Projektleiter, Architekten und Softwareentwickler sich mit etwas beschäftigen, das jenseits von Meilensteinen, Datenbanken, Programmiersprachen und Übertragungsprotokollen liegt: mit dem Benutzer, dem großen Unbekannten, dem wankelmütigen und nach allem verlangenden Kunden. Zumindest in einigen Programmier-Denkstuben herrscht genau diese Vorstellung: Eine Vorstellung, die so nicht stimmt und deren Widerlegung dem einen oder anderen Denkanstöße geben wird, die zu einer besseren User Experience führen.

Wer der Benutzer ist, welche Wünsche und Bedürfnisse er hat und wie man ihn bei der Arbeit oder auch privat am besten unterstützt, darum kümmert sich das *User Centered Design (UCD)*. Wie der Name bereits sagt, ist UCD nicht technologie- oder funktionsgetrieben, sondern konzentriert sich auf den Benutzer. Es verwundert also nicht, dass Methoden, um die Bedürfnisse der Benutzer zu ergründen, ein Kernstück von UCD sind. UCD ist als iterativer Prozess definiert (siehe auch Abbildung 1):

- Die Bedürfnisse der Benutzer werden erhoben.
- Der Benutzungskontext definiert die Rahmenbedingungen, aus denen sich wiederum die Anforderungen ableiten.
- Daraus erarbeitet das Team – je nach Projektphase – *Mockups*, Papier-Prototypen, klickbare Prototypen und schließlich das fertige Produkt.
- Nach jeder Entwicklung muss das Ergebnis wieder geprüft werden.

Schritt für Schritt setzt das Team dabei ein besonderes Puzzle zusammen: Ein möglichst vollständiges und realistisches Bild

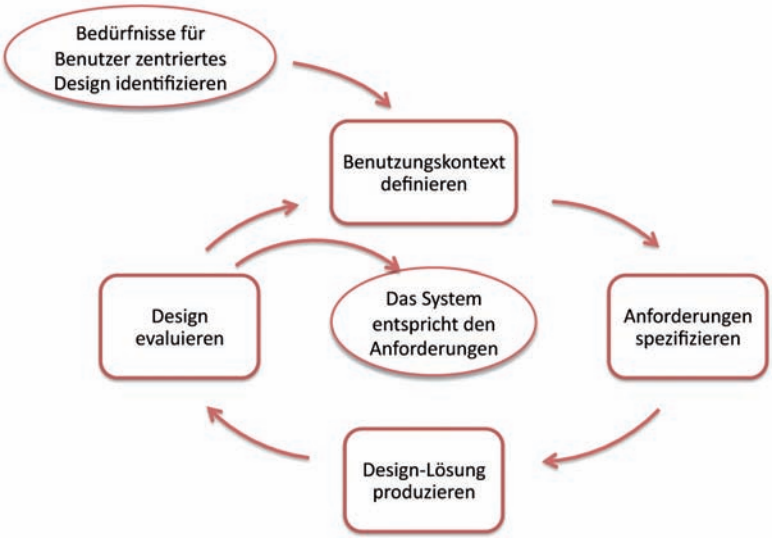

Abb. 1: UCD-Zyklus.

von den Wünschen, Bedürfnissen und Verhaltensmustern des großen Unbekannten, des Benutzers. Denn das ist eine zwingende Voraussetzung für ein Produkt, das nicht einfach nur einem Zweck dient, sondern das heraussticht, Spaß macht oder schlicht „cool" ist.

Der Sprung vom Kopf des Löwen

Den Benutzer ins Zentrum zu setzen, ist also unabdingbar für ein hervorragendes Produkt. Wieso verzichten dann immer noch so viele Projekte darauf? Einige Argumente, die in diesem Zusammenhang häufig genannt werden, sind:

- „Zu teuer."
- „Zu aufwändig."
- „Uns reichen gute Anforderungen."
- „Das Marketing kennt den Benutzer gut genug."
- „Wir wissen selbst, was eine gute Software ausmacht."
- „Wir haben hier kein Wunschkonzert."

Wem kommen solche Aussagen nicht bekannt vor? Natürlich kostet UCD Zeit und Geld, aber wertvolle Dinge gibt es nun einmal selten umsonst. Es bestreitet wohl niemand, dass ein Problem um so teurer wird, je später man es während der Entwicklung entdeckt. *Fail Fast* ist das Motto der meisten modernen Entwicklungsprozesse. Probleme mit der Benutzbarkeit (*Usability*) haben jedoch die hinterhältige Eigenschaft erst aufzutreten, wenn der Kunde das Produkt bereits gekauft hat. Damit gehören sie zu den kostspieligsten Fehlern überhaupt. Und das ist keine neue Erkenntnis.

Viele Organisationen, die den Usability-Aspekt berücksichtigen, rechnen mit einem Kosten-Nutzen Verhältnis von 1 Euro : 10 Euro – 100 Euro. Während der Entwicklung kostet die Behebung eines Problems zehnmal soviel wie während des Designs. Sobald ein System beim Kunden ist, kostet die Behebung sogar hundertmal mehr (siehe Abbildung 2, vgl. [Gil88]).

Bei der Budgetierung von kleinen Projekten erscheint einem der Anteil der Usability häufig untragbar hoch. Für so einen vermeintlich vagen Kostenpunkt eventuell sogar Features zu streichen, ist hart. Ist es in diesem Fall jedoch vernünftig, die Usability in der Kostenschätzung zu ignorieren und auf das Beste zu hoffen? Natürlich nicht. Und trotzdem geschieht

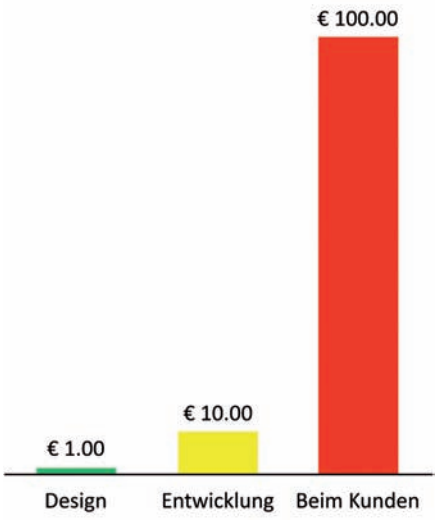

Abb. 2: Relative Kosten von Usability-Problemen.

genau das ständig. Viele vergessen dabei, dass der *UCD*-Prozess (*User Centered Design*) skalierbar ist. Das Projekt muss nicht jede einzelne Methode anwenden, um etwas zu erreichen. Wenig zu tun, ist immer noch um Längen besser, als den Kopf in den Sand zu stecken.

Noch immer ist die Vorstellung von dem allmächtigen Anforderungsdokument, das 100 % des Systems exakt spezifiziert, weit verbreitet. Trotzdem lauten die Anforderungen bezüglich Usability üblicherweise so:

- „Das System soll einfach zu bedienen sein."
- „Das System soll intuitiv anwendbar sein."
- „Die Bedienung soll einfach erlernbar sein."

Wie vielen Testabteilungen haben solche Anforderungen wohl schon Kopfzerbrechen bereitet? Mit UCD entstehen stattdessen überprüfbare Anforderungen, die durch den iterativen Prozess auch regelmäßig getestet werden.

Usablity ist also ein wichtiges Thema. Aber wieso kann der Entwickler eine gute Usability nicht selber gewährleisten? Schließlich arbeitet er jeden Tag mit Software. Da müsste er doch inzwischen wissen, worauf es ankommt. Leider zeigt die Praxis, dass diese Erfahrung die Entwickler im besten Fall dazu befähigt, Software für andere Entwickler zu schreiben. Die tägliche Arbeit eines Entwicklers gibt kaum Aufschluss darüber, was eine Anwältin von einem Produkt zur Suche von Präzedenzfällen erwartet oder was im Kopf eines Künstlers vorgeht, der ohne irgendwelche Vorkenntnisse zum ersten Mal Farben über das Internet bestellen will.

Im Kopf des Benutzers

Wie stellt man ein Produkt her, das den Benutzer begeistert? Agile Methoden wie Scrum haben dafür einen simplen Ansatz: Zukünftige Benutzer werden möglichst eng in den Entwicklungsprozess eingebunden. Sie sind bei der Definition der Anforderungen beteiligt und erhalten in kurzen Iterationen von ein bis vier Wochen lauffähige Versionen, die sie sofort ihren Bedürfnissen gegenüberstellen können. Vor allem in der Software-Dienstleistungsbranche wird dieses Vorgehen sehr erfolgreich eingesetzt.

Die Vorgehensweise der agilen Softwareentwicklung ist zwar sehr effektiv, weist aber eine große Einschränkung auf: Der Benutzer muss verfügbar sein. Da das aber selten der Fall ist, gibt es den *User-Proxy* (siehe Abbildung 3, vgl. [Coh04]) – hierbei handelt es sich um einen Stellvertreter, der während der Entwicklung die Rolle des Benutzers einnimmt. Dieser Stellvertreter muss wissen, wie der Benutzer tickt, und hier kommt der UCD-Prozess ins Spiel. Der UCD-Prozess läuft parallel zum Entwicklungsprozess und füttert den *User-Proxy* fortlaufend mit den nötigen Informationen.

Ermittlungsarbeit

„Wo waren Sie Sonntagabend zwischen 10 und 12 Uhr?" Die Ermittler in den einschlägigen Krimiserien im Fernsehen graben nach Informationen, indem sie Zeugen

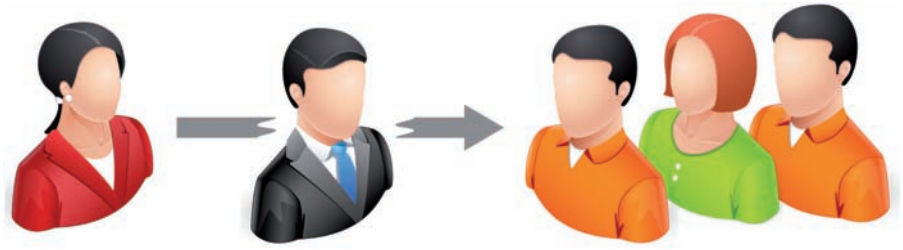

Abb. 3: User-Proxy.

Abb. 4: Interview.

und Tatverdächtige befragen. Dieselbe Methode verwenden auch Usability-Experten, um an Informationen über den Benutzer zu gelangen – auch wenn schlechte Software noch kein offizielles Verbrechen ist.

Das *Interview* (siehe Abbildung 4) mit dem Benutzer liefert Erfahrungen, Erwartungen und Bedürfnisse sowie ein grundlegendes Verständnis für involvierte Personen, Abläufe und Prozesse. Es stellt nur wenige Anforderungen. Der Interviewer muss mit dem Benutzer in Kontakt treten – üblicherweise erfolgt das von Angesicht zu Angesicht, aber auch ein telefonisches Gespräch ist möglich. Das Interview kann beim Kunden, in der eigenen Firma oder an einem andern Ort stattfinden. Firmen, die ein neues Produkt lancieren möchten, versammeln gerne verschiedene potenzielle Kunden und führen Interviews an einem neutralen Ort, wobei der Veranstalter anonym bleibt.

Wie strukturiert die Befragung sein soll, bleibt dem Usability-Experten überlassen. Bietet sich die Gelegenheit für eine große Zahl von Interviews, sollte man einen Fragenkatalog und ein Skript vorbereiten, um vergleichbare Resultate zu erhalten. Der Interviewer kann immer noch spontan mit gezielten Fragen Themen weiter ausleuchten (*Probing*). Falls das Ihr erstes Interview ist, sollten Sie auch bei wenigen Interviewpartnern einen Fragenkatalog erstellen. Ansonsten gehen wichtige Punkte aus lauter Nervosität verloren.

Was jeder Ermittler wissen muss, ob nun im Fernsehen oder für UCD: Das Ergebnis eines Interviews ist immer die persönliche Sicht des Befragten. Aussagen über Performance oder Einfachheit der Benutzung sind mit Vorsicht zu betrachten. Hollywood führt den Zuschauer gern in die Irre, indem Zeugen ihre persönliche Wahrheit berichten. Nicht selten führt erst der Videobeweis zurück auf die richtige Spur.

Observierung

Zwei gelangweilte Beamte, nachts, in einem unauffälligen Fahrzeug, der eine mit Fernglas, der andere isst gerade Hühnchen vom chinesischen Schnellimbiss um die Ecke. Dieses Bild drängt sich bei dem Begriff Observierung auf. Das Beobachten ist zwar auch ein zentraler Bestandteil von UCD, läuft aber etwas weniger dramatisch ab.

- Bei der *Beobachtung* (siehe Abbildung 5) begleitet der Usability-Experte den Benutzer bei Arbeitsschritten oder ganzen Arbeitstagen, ohne jedoch selbst einzugreifen. Falls erlaubt, kann man am entsprechenden Arbeitsplatz eine Kamera installieren und die Tätigkeiten filmen. Sehr sinnvoll ist dies an hektischen Orten, wie zum Beispiel in einer Bäckerei, wo der Beobachter nur im Weg stehen würde.
- Die Kontextanalyse (*Contextual Inquiry*) (siehe Abbildung 6, vgl. [Bey97]) ist eine interaktive Form der Beobachtung. Dabei geht man an den Arbeitsplatz des Benutzers und befragt ihn zu seiner Arbeit, den Abläufen sowie seinen Werkzeugen. Angestrebt wird damit ein Meister-Lehrling-Verhältnis, um die Geheimnisse des Arbeitsalltags zu erfahren. Der Designer wird dabei quasi zum Benutzer ausgebildet.

Personen, die eine *Contextual Inquiry* durchgeführt haben, eigenen sich bestens als *User-Proxy* für die Projektteams. Im Idealfall haben sie die Sorgen und Schwierigkeiten des Benutzers sogar am eigenen Leib erfahren. Obwohl Beobachtung und *Contextual Inquiry* um einiges aufwändiger sind als das Interview, liefern sie häufig die entscheidenden Informationen, um das Produkt drastisch zu verbessern. Beschreibt ein Benutzer seine Arbeitsweise in einem Interview, erwähnt er üblicherweise nur die bewussten Handlungen und auch nur diejenigen, an die er sich gerade erinnert (siehe Kasten 1).

Groß denken

Das Ganze ist größer als die Summe seiner Teile. Sammelt man Informationen immer von einzelnen Benutzern, kommt es oft vor, dass gewisse Dinge in den Arbeitsabläufen nicht angesprochen und so auch nicht erfasst werden. In einer *Group Task Analysis* (vgl. [Cou05]) setzen sich Benutzer zusammen, um gemeinsam einen Ablauf in seiner gesamten Größe zusammenzutragen und sogar zu dokumentieren.

Eine Aufgabenanalyse ist als typischer Workshop angelegt. In Kleingruppen tragen die Teilnehmer auf selbst geschriebenen Karten die einzelnen Schritte eines Arbeitsablaufs zusammen. Zusätzlich zum endgültigen Modell sollte der Prozess gefilmt werden, um wichtige Gedanken, die während der Diskussion wieder weggefallen sind, zu erhalten.

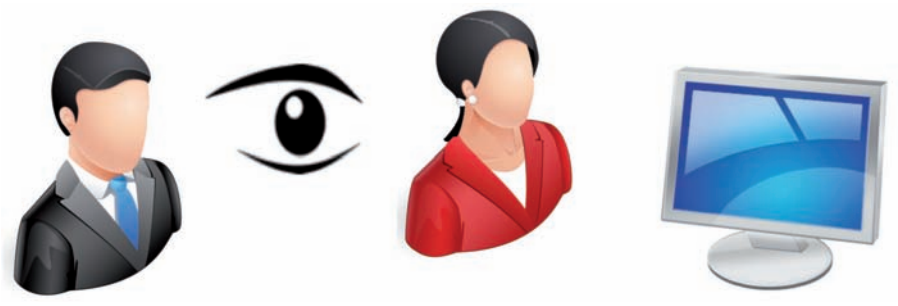

Abb. 5: Beobachtung.

Abb. 6: Contextual Inquiry.

Noch größer

In der vernetzten Welt haben selbst kleinste Softwareprodukte ohne viel Aufwand die Chance auf Millionen von potenziellen Benutzern. Für lokalisierte Software ist das Publikum sogar noch größer. Mit Interviews und Beobachten kommt man da nicht weit. Nun rücken die quantitativen Daten mehr in den Vordergrund. Wie viele Benutzer sind Anfänger und wie viele Experten? Zu welchem Zweck verwenden sie das Produkt vorwiegend? Wie viele von ihnen lesen Bedienungsanleitungen?

Hier kommt der *Fragebogen* (siehe Abbildung 7, vgl. [Cou05]) zum Einsatz. Er adressiert ein großes Publikum und erlaubt eine statistische Auswertung der Resultate. Selbst bei Produkten, die nicht gleich auf die ganze Welt abzielen, hat er seine Vorteile:

- Die Befragten können selbst den Zeitpunkt wählen, zu dem sie ihre Antworten geben.
- Während der Beantwortung besteht kein Zeitdruck.
- Die Befragten können anonym bleiben.

Auf den ersten Blick scheinen Versand und Auswertung den Löwenanteil der Arbeit auszumachen. In Wahrheit ist jedoch das Formulieren der Fragen der aufwändigste Schritt. Gute Fragen produzieren gute und weniger gute Antworten. Schlechte Fragen produzieren, wenn überhaupt, nur wertlose Antworten. Gute Fragen bzw. gute Fragebogen erfüllen zwei wichtige Kriterien: Die Antworten lassen sich vernünftig auswerten und in Bezug zum Produkt stellen. Und die Befragten werden genügend motiviert, um nicht schon nach der zweiten Frage aufzugeben.

Daten, Daten, Daten

Auf den Schreibtischen des UCD-Teams stapeln sich die Notizen von Interviews, DVDs mit Videoaufnahmen und zurückgesandte Fragebögen. Das Team hat sie gefiltert, priorisiert, Muster isoliert und Statistiken erstellt. Wie geht es jetzt weiter und was war eigentlich das Ziel? Das Entwicklerteam wollte einen Benutzer als Teammitglied – falls möglich, sogar einen für jede Kategorie von Benutzern, die das Produkt verwenden. Mit all den Daten müsste es doch möglich sein, den Entwicklern etwas Vergleichbares zu bieten.

Eine Angestellte kopiert für einen Arbeitsschritt Daten von einer Arbeitsstation von Hand auf einen Notizzettel und überträgt diese dann auf einer anderen Arbeitsstation in das Produkt, das unter Beobachtung steht. Bei der Auswertung stellt man fest, dass die erste Arbeitsstation einen Export der Daten unterstützt. Mit einem entsprechenden Import im untersuchten Produkt könnte man den Arbeitsschritt automatisieren. In einem Interview hat die Angestellte diesen Arbeitsschritt aber nicht erwähnt. Vielleicht war er ihr nicht bewusst, vielleicht war es ihr peinlich oder vielleicht erschien ihr eine Erwähnung irrelevant, da sie die technischen Möglichkeiten nicht kannte. Wenn man nur beobachtet, kann man den Grund ebenfalls nicht erfragen. Die größte Chance, sowohl den Arbeitsschritt als auch die Beweggründe und Hintergedanken zu erfahren, hat man mit einem *Contextual Inquiry*.

Kasten 1: Vorteil von Contextual Inquiry an einem Beispiel.

Der Benutzer-Klon

Die DNA des Benutzers ist entschlüsselt. Der nächste Schritt besteht darin, einen Benutzer-Klon erstellen, oder – weniger salopp und dafür fachlich präziser – eine *Persona*. Personas sind Dokumente, die die typischen Benutzer des Produkts steckbriefartig beschreiben (siehe Abbildung 8). Der UCD-Experte erstellt aus den gesammelten Daten fiktive Benutzer, mit denen er im weiteren Verlauf des UCD-Prozesses arbeiten kann.

Die Bedeutung der Personas für das Entwicklerteam wird häufig unterschätzt. Für das Team ist eine Persona der Ersatz für einen echten Benutzer. Darum sind die Entwickler vor allem an produktrelevanten Informationen interessiert. Dass eine Persona zwei Kinder hat, jeden Samstag einen Yoga-Kurs besucht und schon einmal auf dem Eiffelturm war, macht sie zwar interessanter, löst bei den Entwicklern jedoch nur ein müdes Lächeln oder Kopfschütteln aus, wenn es sich bei dem Produkt z. B. um ein Diagnosegerät in einem Labor handelt, in dem die Persona arbeitet.

Eine gut geschriebene Persona enthält alle relevanten Informationen aus den gesammelten Daten, die Auskunft geben über Probleme, Motivation, Bedürfnisse, Wünsche, Fähigkeiten, Verhaltensmuster – ja sogar Ängste – der realen Benutzer. Ein Entwickler kann eine solche Persona „fragen", was sie möchte und ob sie mit den Ergebnissen zufrieden ist. Eine Persona kann in fünf Komponenten unterteilt werden:

- *Hintergrundinformationen:* Ausbildung, Alter, soziales Umfeld.
- *Charakter:* Technikaffin, zurückhaltend gegenüber Neuerungen, methodisch.
- *Ziele & Motivation:* Weniger Papierarbeit, mehr Zeit für Kundenbetreuung.
- *Ärger & Ängste:* Mehrere Log-Ins, Fehler können nur vom Administrator behoben werden.
- *Szene:* Die Persona erzählt über sich und ihren Alltag und vermengt so die Punkte 1-4.

Die Auflistung ist nicht abschließend und kann in diesem Sinne noch durch eigene

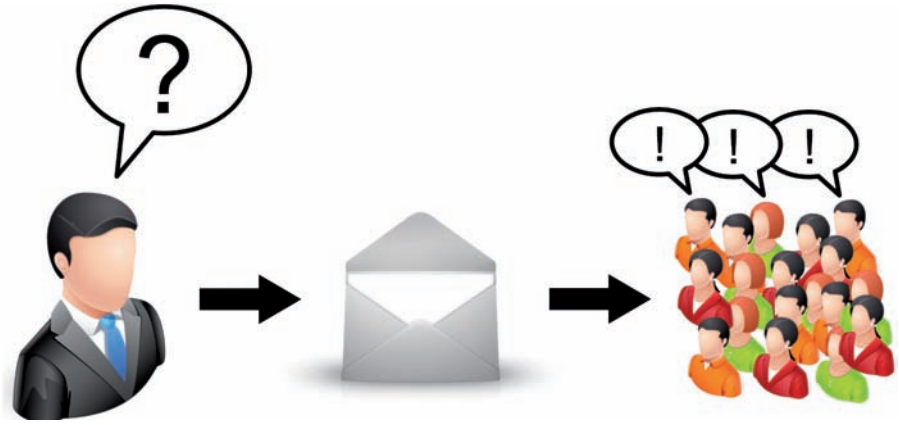

Abb. 7: Fragebogen.

Ideen erweitert werden. Weitergehende Informationen finden sich in [Cop07].

In Szene setzen

Personas allein wären wie ein Buch, in dem der Autor nur die Personen beschreibt, oder ein Actionfilm, in dem sich niemand bewegt: Die Handlung fehlt. Dazu entwirft das UCD-Team Szenario-Dokumente. Szenarien werden in der Softwareentwicklung häufig mit Use-Cases verwechselt. Im Gegensatz zu Use-Cases erzählen Szenarien aber Geschichten, bei denen nicht das Produkt im Zentrum steht, sondern in denen konkrete Personen und alltägliche Situationen eine wichtige Rolle spielen. Ein Szenario sollte geschrieben sein wie ein gutes Buch oder zumindest die Verfilmung davon. Zweck eines Szenarios ist es, einerseits die Arbeitswelt, Umgebungsbedingung und Interaktionen mit anderen Personen zu dokumentieren. Andererseits sind Szenarien eine große Hilfe, um das Produkt bei Projektleitern und Entwicklern in die richtige Perspektive zu rücken.

Kasten 2 zeigt, wie einfach ein Szenario aussehen kann. Trotzdem muss darauf geachtet werden, dass man sich auf den Einsatz des Produkts konzentriert – so wie in einem Krimi immer der Kommissar und der Täter im Mittelpunkt stehen. Szenarien helfen nicht nur den UCD-Experten, selbst Entwickler können neue Erkenntnisse daraus ziehen oder bei Lücken in den Spezifikationen Lösungen im Sinne des Anwenders vorschlagen.

Iterativ

Nach dem Erstellen von Personas, Szenarien und eventuell noch anderen Dokumenten wie Benutzungsschnittstellen-Guidelines ziehen viele Firmen, die UCD nur halbherzig betreiben, bereits den Schlussstrich. Möglicherweise hat ein UCD-Experte am Ende des Projekts noch Zeit für ein Review, falls man Glück hat.

In Wirklichkeit ist UCD aber ein iterativer Prozess. Und das Projekt steht gerade erst vor dem Ende der ersten Iteration. Doch bevor das Team mit der nächsten Iteration beginnen kann, muss es das Erreichte prüfen, um festzustellen, was noch zu tun ist. Wie das geschieht, hängt vom Zeitpunkt ab. Zu Beginn des Projekts erstellte das UCD-Team häufig Papier-Prototypen. Diese sind günstig und können mit ausgewählten Benutzern einfach durchgespielt werden. Ab einer bestimmten Projektgröße erstellt das Entwicklerteam aus den Ergebnissen zusätzlich einen richtigen Prototypen, mit dem die Benutzer zusammen mit dem UCD-Team die wichtigsten Szenarien überprüfen können.

Bei herkömmlichen Projekten folgt darauf eine Durststrecke, bis sich die Software ihrer Vollendung nähert. Erst dann beginnt erneut eine UCD-Iteration, diesmal mit dem echten Produkt. Nicht so bei agilen Methoden, die in kurzen Iterationen theoretisch auslieferbare Software produzieren. In diesem Fall kann das UCD-Team im gleichen Rhythmus – versetzt zum Entwicklerteam – seine Iterationen fortsetzen.

Ab ins Labor

Wenn in einem Labor an Frankensteins Schöpfung gearbeitet wird, sehen wir alle gebannt auf die Kino-Leinwand. Doch wie schafft es der Regisseur, den Geschmack seines Zielpublikums so genau zu treffen? In den Traumfabriken Hollywoods wird kaum ein Film veröffentlicht, ohne dass dieser von einem Testpublikum begutachtet wurde. So etwas wird auch bei Geräten oder Programmen gemacht. Es ist sozusagen die Königsdisziplin der UCD-Methoden: das *Usability-Lab*. Hier lösen Benutzer Aufgaben mit ersten Prototypen oder dem fertigen Produkt.

Ein typisches Usability-Lab besteht aus zwei Räumen (siehe Abbildung 9). Im ersten Raum ist ein Arbeitsplatz für die Probanden eingerichtet. Im zweiten sitzen die Beobachter und der Testleiter. Auf den in jedem Krimi benutzen Einwegspiegel sollte man aus psychologischen Gründen verzichten. Stattdessen beobachten die Experten mit kleinen Kameras und passender Software die Probanden und ihre Arbeit.

Der Testleiter gibt dem Benutzer verschiedenste Aufgaben, die es zu lösen gilt. Die Beobachter achten auf alle Aktionen und Reaktionen seitens des Benutzers. Wenn der Proband dazu noch laut denkt, was er gerade tun will, führt das zu Erkenntnissen, die selbst dem größten Usability-Skeptiker beweisen, dass ein schlecht designtes Produkt weltweit nur

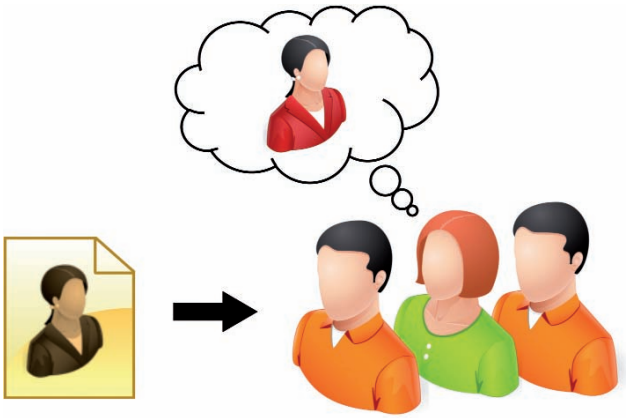

Abb. 8: Personas.

Torsten Müller (Wartungstechniker) sieht auf seinem Laptop, auf dem die Software installiert ist, nach, welche Aufträge heute noch anstehen. Er wählt den nächsten Auftrag an und die Reparaturen der letzten Monate erscheinen nebst einem Hinweis, dass der Zahnriemen ersetzt werden muss und der Kunde wegen eines vermuteten Wackelkontakts angerufen hat. Torsten bestellt einen Riemen sowie ein neues Bedien-Panel per Knopfdruck, sodass er es auf dem Weg zu seinem Dienstwagen gleich mitnehmen kann.

Kasten 2: Beispiel-Szenario.

von seiner Entwicklungsabteilung bedient werden kann.

Doch seien Sie gewarnt: Die Planung, Durchführung und Auswertung eines Usability-Lab ist ziemlich aufwändig und schwierig. Überlassen Sie solche Aufgaben den Experten, denn es wäre wirklich schade um Zeit und Geld, wenn nachher unbrauchbare Resultate erzielt werden. Richtig durchgeführt, erspart es aber unnötige Reklamationen von Kunden, umfangreiche Bedienungsanleitungen, schlechte Kritik in den Medien und den daraus folgende Imageschaden. Daher gibt Hollywood selten Filme heraus, die komplett unverständlich sind.

Den Experten fragen

Selbst Großkonzerne mit eigenen Usability-Labs testen gewöhnlich nicht jede Iteration des Entwicklerteams im Labor. Die Entwickler müssen aber an einem Iterationsende nicht nur feststellen, ob alle geplanten Anforderungen umgesetzt sind, sondern auch, ob das Produkt den Usability-Ansprüchen genügt. Während der Entwicklung verwenden sie Personas und Szenarien als Ersatz, für Tests benötigen sie aber echte Personen. Es liegt nahe, dieselben Personen einzusetzen, die auch die Anforderungen testen. Das ist aber nicht die beste Wahl, denn die Tester beziehen ihre Informationen zu den Usability-Anforderungen aus denselben Dokumenten wie die Entwickler. Alles, was sie also bezüglich Usability beisteuern können, ist eine unabhängige Meinung.

Das UCD-Team hat andererseits intensiven Kontakt zum Kunden. Es sind seine Erfahrungen und Erkenntnisse, aus denen die Personas und Szenarien entstehen. Statt also mit dem Kunden selbst nach einer Iteration das Produkt zu prüfen, was die beste Wahl wäre, aber eben selten möglich ist, übernimmt der verantwortliche UCD-Experte diese Rolle und macht mit dem Entwicklerteam einen Experten-Walkthrough. Das Ergebnis dient als Input für die nächsten Iterationen, sowohl des UCD als auch des Entwicklerteams.

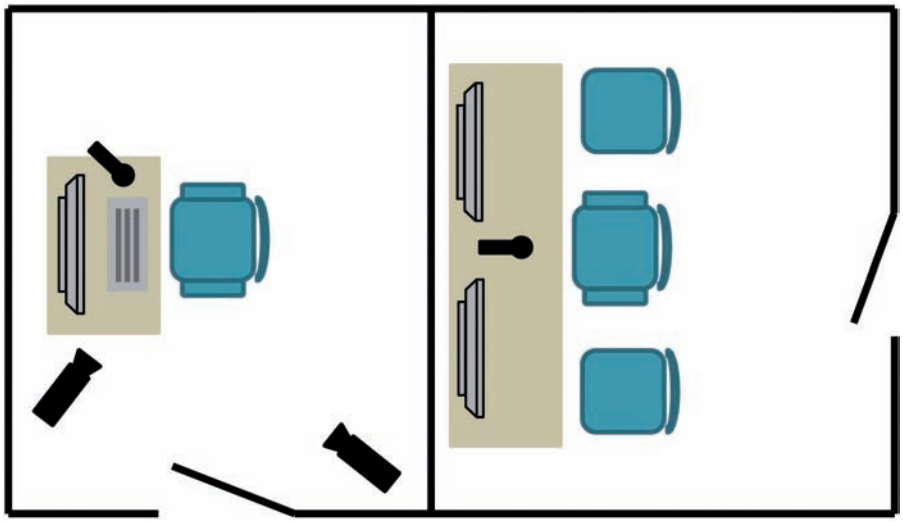

Abb. 9: Grundriss eines Usability-Labs.

Schöne neue Welt

Fragen, Beobachten, Auswerten, Prototypen bauen, Benutzer klonen, im Labor testen und das ganze wiederholen, bis am Ende ein Produkt entsteht, das sich mit iPhone & Co. messen kann – das klingt nicht nur nach einer Menge Arbeit, es ist es auch. Kleine Projekte können einen umfassenden UCD-Prozess kaum stemmen. Zum Glück ist das auch nicht nötig, denn nur wer sich mit Apple messen will, benötigt vergleichbare finanzielle Mittel.

Ein Kleinprojekt erreicht schon viel, indem es jemanden kurze Zeit beim Kunden arbeiten lässt (*Contextual Inquiry*), der dann als Stellvertreter für den Benutzer fungiert (*User-Proxy*) und regelmäßig das Produkt bezüglich Usability überprüft (*Experten-Walkthroughs*). Falls das Projekt den UCD-Prozess nicht selbst hochziehen will, gibt es auch die Möglichkeit, externe Partner hinzuzuziehen, die sich auf das Usability-Engineering spezialisiert haben. Außerdem ist es bei Kleinprojekten am wahrscheinlichsten, dass der Benutzer bereit ist, selbst am UCD teilzunehmen – ein Vorteil, auf den Großprojekte meist verzichten müssen.

Ein letzter Blick

UCD rückt den Benutzer ins Zentrum. Es kostet Zeit und Geld, ist aber auf die Projektgröße adaptierbar. Es gibt dutzende weiterer Methoden, deren Erklärung den Rahmen dieses Artikels sprengen würde. Aus diesen kann sich das Projektteam diejenigen heraussuchen, die den Zielen und dem Budget des Projekts entsprechen.

UCD allein garantiert keinen Erfolg. Moderne Technologien, eine große Portion Kreativität und etwas Glück gehören ebenfalls zum Rezept. Trotzdem ist UCD eine unverzichtbare Zutat für ein Produkt, das nicht einfach nur einem Zweck dient, sondern das heraussticht, Spaß macht oder schlicht „cool" ist. ■

Literatur

[Bey97] H. Beyer, K. Holtzblatt, Contextual Design – Defining Customer-Centered Systems, Morgan Kaufmann 1997

[Coh04] M. Cohn, User Stories Applied, Addison-Wesley 2004

[Cop07] A. Cooper, About Face 3 – The Essentials of Interaction Design, John Wiley & Sons 2007

[Cou05] C. Courage, K. Baxter, Understanding Users – A Practical Guide to User Requirements Methods, Tools & Techniques, Morgan Kaufmann 2005

[Gil88] T. Gilb, Principles of software engineering management, in: Usability Is Good Business, 10/2001

[Nor02] D.A. Norman, The Design of Everyday Things, Perseus Books 2002