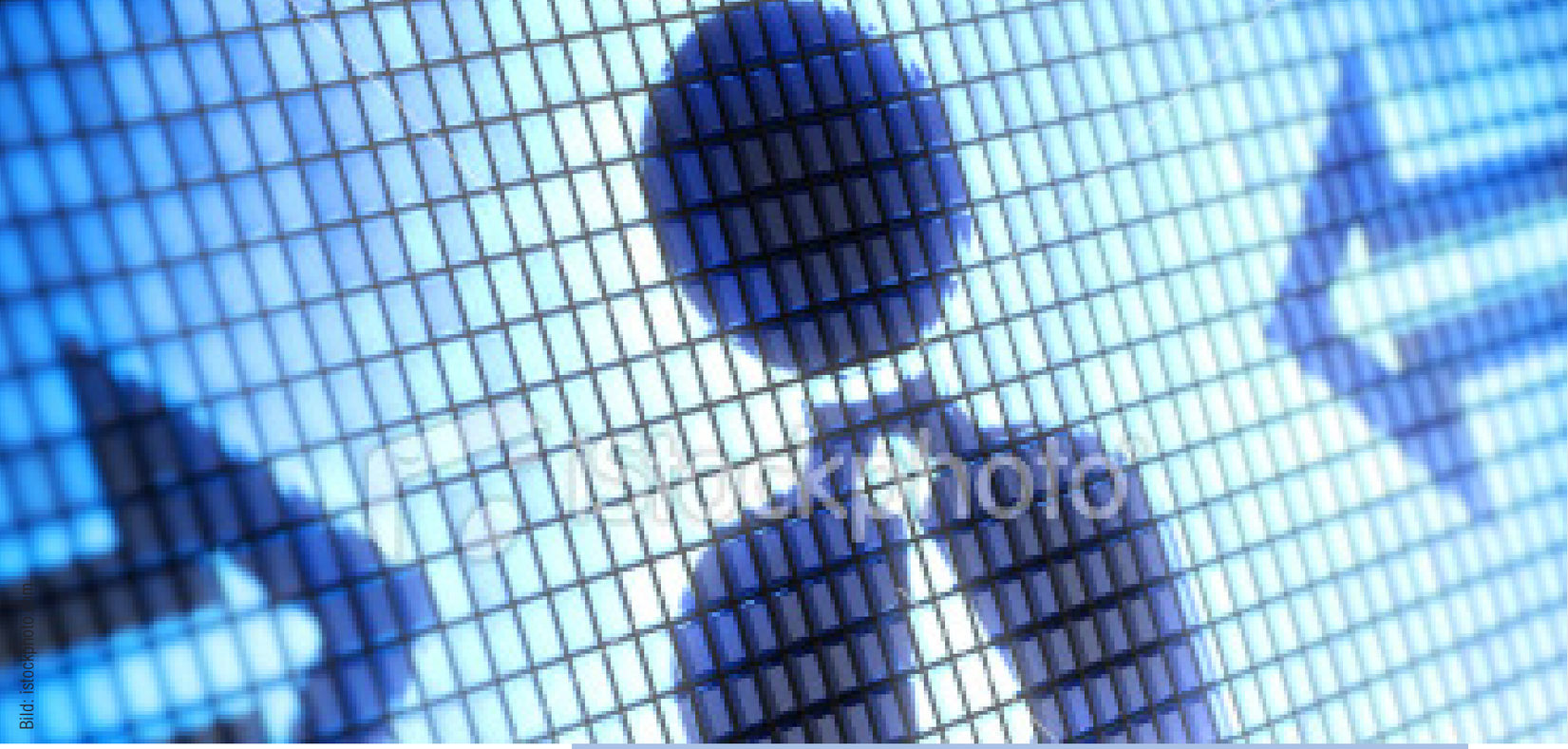


Bild: istockphoto.com

DOSSIER USER-CENTERED DEVELOPMENT
IN KOOPERATION MIT: DIE ERGONOMEN USABILITY AG

Entwickler und User vereint

jae. Bei der Softwareentwicklung ist es wie auch sonst im Leben. Wer nicht miteinander spricht, wurstelt vor sich hin, Missverständnisse sind vorprogrammiert und der Frust am Ende ist meist gross.

Software soll einfach sein, sie soll den Nutzer nicht überfordern und sie soll ihm vor allem das Leben einfacher und nicht schwieriger machen. Leider scheint sich diese Ansicht noch nicht überall in der IT-Branche durchgesetzt zu haben. Denn noch immer gibt es frustrierte Nutzer, die sich mit einer Software nicht verstehen. Wobei das vielleicht nicht immer nur an der Software alleine liegt.

Zumindest kann es nicht schaden, den Nutzer bei der Entwicklung von Software miteinzubeziehen. Vor allem dann, wenn man verhindern will, dass kleine wie grosse IT-Projekte zum Albtraum werden. Dabei darf man die Entwickler nicht vergessen. Bringt man sie mit den Nutzern zusammen, damit sie deren Sprache und Vorstellungen besser verstehen können, wissen sie auch eher, wie das Endprodukt aussehen und funktionieren soll.

Um die Kommunikation zwischen Entwicklern und Nutzern und in den Projektteams zu unterstützen, stehen heute viele Hilfe-Tools zur Verfügung. Es gibt aber auch die Möglichkeit, einen intelligenten Mock-up zu erstellen oder den Nutzern einen Prototypen zu präsentieren, den sie testen können, wie Leon V. Schumacher im Interview auf Seite 40 sagt. Sind die Nutzer mit dem Resultat nicht zufrieden, müssen die Entwickler wieder ran, die Armen.

Auf diese Weise kann zumindest theoretisch nicht mehr allzu viel schiefgehen. Es sei denn, die Nutzer haben unrealistische Vorstellungen des Endprodukts, oder den Entwicklern stellen sich unüberwindbare technische Schwierigkeiten in den Weg. Letzteres lässt sich dann allerdings auch nicht durch miteinander Reden aus dem Weg schaffen. <

> **Seite 38**
Mind the Gap! Wie Schnittstellen zu Schnittmengen werden

> **Seite 40**
Leon V. Schumacher, Baxian: «Viele Entwickler bleiben sehr weit von der Sichtweise des Benutzers entfernt»

Mind the Gap! Wie Schnittstellen zu Schnittmengen werden

Früher reichte es, programmieren zu können, um eine gut verkaufbare Software zu entwickeln. Heutzutage ist die Konkurrenz gross, die Benutzer anspruchsvoll. Qualität und Bedienbarkeit sowie das Look & Feel der Oberfläche werden immer wichtigere Verkaufsargumente. Neue Modelle und Herangehensweisen sind gefragt. Philipp Klett, Christopher H. Müller

Der aktuelle Prozess der Softwareentwicklung lässt sich aktuell grob in vier Phasen unterteilen: Analyse, Konzeption, Entwicklung und Einführung. In jeder Phase werden spezielle Werkzeuge verwendet, um ein für die entsprechende Phase und aus der Sicht des Spezialisten möglichst optimales Ergebnis zu erhalten. Durch diese klare Trennung der Phasen entstehen sogenannte «Gaps» – nicht klar definierte Übergabe-Bereiche zwischen zwei Phasen. Jeder Gap ist eine potenzielle Fehlerquelle. Am Ende einer Phase werden die Ergebnisse in die nächste Phase übergeben: 1.) Analyse-Spezialisten dokumentieren den Ist-Zustand und erheben Anforderungen für das neue Projekt – das Ergebnis sind Anforderungsdokumente. 2.) Konzepter entwickeln aus den Anforderungen ein Konzept – daraus ergibt sich das Konzept beziehungsweise ergeben sich Konzeptvarianten. 3.) Entwickler setzen das Konzept auf der Zielplattform um, aus Papier wird digitaler Code – das Softwaresystem ist komplett. 4.) Schulungsspezialisten und Support sorgen für eine reibungslose Einführung der neuen Software – das Resultat sind zufriedene, effiziente Nutzer der Software.

Wo so viele verschiedene Menschen mit unterschiedlichen Werkzeugen, Ansichten, Denk- und Arbeitsweisen aufeinandertreffen und zusammenarbeiten müssen, kommt es zwangsläufig zu (Übersetzungs-)Fehlern. Das Ergebnis aus der vorangegangenen Phase muss immer interpretiert werden. Gründe dafür: Die verwendeten Hilfsmittel passen nicht zusammen; das Verständnis für die Tätigkeit der Personen in den anderen Phasen ist gering; es werden unterschiedliche Sprachen verwendet (Phase 2: Bildsprache <> Phase 3: Programmiersprache)

Philipp Klett ist Senior Usability-Engineer und User Experience Consultant mit mehreren Jahren Entwickler-Erfahrung.
Dr. Christopher H. Müller ist Inhaber und CEO von Die Ergonomen Usability AG aus Zürich.

User Experience Design / Development

Was in der Konzeptphase entworfen, visualisiert und evaluiert worden ist, muss in ein funktionierendes und intuitiv benutzbares System umgesetzt werden. Normalerweise werden dazu auf der Basis von Mock-ups aus der Konzeptionsphase Prototypen entwickelt und evaluiert. Parallel beginnt das Grafik-/Screendesign mit der Gestaltung und Ausarbeitung von Icons, Logos oder Farbkonzept. Werden bei der Evaluation die Usability-Kriterien erfüllt, werden die verschiedenen Prototypen an die Entwickler übergeben und von diesen im finalen System umgesetzt.

Bei der Umsetzung müssen sie dabei unter anderem Style-Guides, visuelle Prototypen, Skizzen, Dokumentationen und Ablaufdiagramme beachten, denn die Informationen über das endgültige, gewünschte Ergebnis liegen in allen diesen Dokumenten verstreut. Als würde das nicht reichen, gibt es meist noch mündliche Absprachen und Änderungen zu beachten, die man in der Kaffeeküche oder auf dem Gang besprochen hat, nicht dokumentiert natürlich. Ziemlich sicher ist dann auch die Dokumentation nicht auf dem neusten Stand.
Es existiert meist eine klare Trennung zwischen den Phasen Konzeption und Entwicklung. Ergebnisse aus der einen Phase werden mit mehr oder weniger geeigneten Hilfsmitteln in die nächste Phase übergeben und dort von anderen Personen weiterverarbeitet. Das ist nicht effizient. Wissen und Erfahrungen aus der Arbeit in einer Phase können oft nicht dokumentiert werden und geraten dadurch in Vergessenheit. Die Folge: Es gibt Redundanzen und Fehler werden wiederholt.

Die Lösung

Für einen möglichst reibungslosen Ablauf der Übergabe empfiehlt es sich dringend: 1.) Prozesse und Übergabeparameter (Schnittstellen) genau zu definieren. 2.) Personen zu integrieren, die mehrere «Sprachen» sprechen und phasenübergreifend agieren können. 3.) Verständnis für andere Fachbereiche zu schaffen, also Hand in Hand statt gegeneinander zu arbeiten. 4.) Werkzeuge und Hilfsmittel anzugleichen und wo möglich zu reduzieren und 5.) Phasenübergreifende Softwareentwicklung zu leben.

Ziel ist es, die Anzahl Schnittstellen und damit Gaps zu verringern und zu überbrücken, Kompetenzen zu bündeln, und vorhandenes Wissen und Erfahrungen nahtlos weiterzureichen. Wenn das gelingt, dann resultieren daraus eine effizientere Vorgehensweise und ein besseres Endprodukt.

Integrierte GUI-Entwicklung

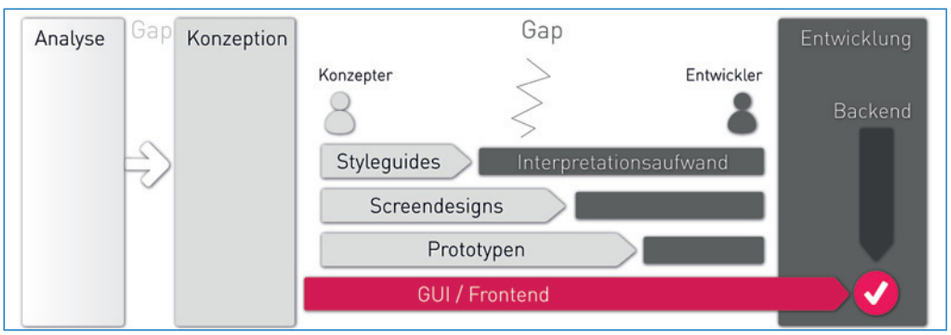
Durch integrierte GUI-Entwicklung wird der Gap zwischen Konzeption und Entwicklung effektiv verkleinert. Auf die üblichen Werkzeuge wie Storyboards, Prototypen etc. wird nicht verzichtet. Sie werden in reduziertem Umfang im Rahmen der GUI-Entwicklung eingesetzt, um komplexe Features mit Benutzern evaluieren zu können. Wo sinnvoll und so früh wie möglich wird aber schon mit der Umsetzung der Konzepte begonnen. Anstatt in der Konzeptphase Zeit für aufwendiges Erstellen von Dokumentation, Style-Guides und grafischen Prototypen aufzuwenden, wird die Entwicklung komplett funktionierender und benutzerfreundlicher User Interfaces direkt auf der Zielplattform realisiert.

Beim Einsatz agiler Methoden wie Scrum kann ein solcher Prozess in zeitlich vorgelagerten User-Experience-Sprints (UX-Sprints) umgesetzt werden. UX geht also in Vorleistung und entwirft eine (Diskussions-)Grundlage für den Entwicklungs-Sprint. So kann ein daraus resultierendes funktionierendes GUI auf der Zielplattform schon sehr früh realisiert, ausgebaut und ins Projekt mit eingebunden werden. Regelmässig wird der aktuelle Entwicklungsstand mittels Usability-Test oder Experten-Evaluation aus Benutzersicht auf Herz und Nieren geprüft. Diese Inputs

fliessen wiederum in die nächsten UX- und Entwicklungs-Sprints mit ein.

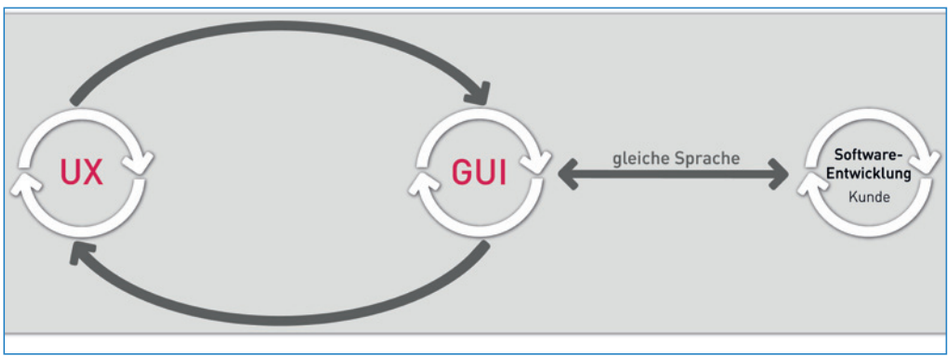
Die Vorteile

Eine enge Zusammenarbeit zwischen Konzeptern, User Experience (UX) Experten und Entwicklern reduziert Missverständnisse, der Koordinationsaufwand wird gesenkt und die GUI-Entwickler werden direkt entlastet. Iteratives User Centered Frontend Design senkt die Projektrisiken und erhöht die Usability und damit die Akzeptanz des Endprodukts, und die laufende Überprüfung der Machbarkeit von Design und Bedienkonzept deckt Probleme frühzeitig auf.



Direkte Umsetzung des Frontends – anstatt Interpretation der Konzept-Outputs durch Entwickler.

Bild: Die Ergonomen Usability AG



Klare Schnittstellen zwischen Usability/Frontend-Team und Kunde. Bild: Die Ergonomen Usability AG

Entwicklern wird die mühsame und unbeliebte Frontend-Design-Arbeit abgenommen, sie müssen die fertigen Screens und/oder Elemente, im Optimalfall die komplette View nur noch mittels zuvor gemeinsam definierter (oder vorgegebener) Schnittstellen an Model und Controller anbinden. Dieses Model-View-Controller-Vorgehen (MVC) wird unter anderem von Microsoft schon seit längerem propagiert. Die Softwaremodule Model (Daten), View (Oberfläche) und Controller (Funktionalität) werden dabei klar voneinander getrennt entwickelt und sind so flexibel und einfacher austauschbar. Beispiel: Die Oberfläche einer Desktop-Applikation kann einfach durch eine spezielle Mobile-View ersetzt werden, die Funktionalität und die Daten bleiben gleich. Durch Weiterverwendung dieser konsistenten Elemente kann man so mit geringem Aufwand eine mobile Applikation, beispielsweise für Windows Phone 8, entwickeln.

Ein Beispiel aus der Praxis

In einem aktuellen Projekt wurde für eine browserbasierte Applikation (IE8) auf SAP-Basis eine komplette Oberfläche in ASP.NET unter anderem mit Web Forms erstellt. Dazu wurden Prototypen zur schnellen Visualisierung und effizienten Abstimmung zwischen Auftraggeber, Design/UX und Entwickler verwendet. Alle benötigten Grafiken und Icons wurden in optimaler Qualität erstellt und direkt an der richtigen Stelle platziert. Die so erstellten Screens wurden von den Entwicklern eins zu eins ins bestehende System integriert. Dabei wurden mit Dummy-Daten und –Schnittstellen zur Datenanbindung gearbeitet, CSS zur Gestaltung der Oberfläche (zum Beispiel autoskalierende Tabellen) und Javascript für Interaktivität und Effekte (zum Beispiel Onscreen-Touch Keyboard). Der letzte Schritt der Integration der Oberfläche ins aktuelle System inklusive SAP-Datenanbindung ging sehr schnell und reibungslos vonstatten.

Im gleichen Zuge wurde eine mobile Variante für einige Funktionalitäten erstellt. Diese sollte auf industrietauglichen Handhelds mit geringer Auflösung im vorinstallierten Browser IE6 Mobile laufen, aber gleichzeitig auch auf modernen Smartphones optimal bedienbar sein. Dank Prototyping auf der Plattform (erst mit jQuery Mobile, was dank fehlender Browser-Unterstützung scheiterte, letztlich dann mit «altmodischem» HTML, JavaScript und CSS) konnte iterativ und zeitnah Feedback eingeholt und so die gewünschte Richtung der Entwicklung effizient gesteuert werden.

Oft hat es Sinn, einen oder sogar zwei Schritte weiter zu gehen. Nicht nur Dokumente zu produzieren, die das Ergebnis möglichst genau beschreiben sollen und doch nicht können, sondern usability-optimierte GUIs zu entwickeln. Mit dem Benutzer-Input aus der Evaluation im Hinterkopf. Aus einer Hand direkt auf der Zielplattform entwickelt.

Entwickler sind Experten

Sie besitzen wertvolles Fachwissen auf ihrem Gebiet und programmieren routiniert die genialsten Tools. Vielen Experten ist aber oft eine gewisse Betriebsblindheit gemein. Der Softwaremarkt ist im Wandel: Es reicht nicht mehr, eine Software nur zu programmieren und dann den Benutzer seinem Schicksal zu überlassen. Um erfolgreich Software entwickeln zu können, müssen heute viele Disziplinen und unterschiedliche Persönlichkeiten zusammenarbeiten. Von Projektmanagern und Marketingfachleuten über Designer und Entwickler bis hin zum Endkunden. Hier ist es wichtig, den Blick von aussen zuzulassen – durch Usability-Experten und vor allem durch Benutzer! Mit möglichst geringem Zeit- und Kostenaufwand sollen möglichst viele Fehler gefunden und ausgemerzt werden. Hierin haben Usability-Experten grosse Erfahrung: Durch die richtige Auswahl an Werkzeugen, Methoden und Testpersonen kann genau das erreicht werden: Eine möglichst fehlerfreie und gut zu bedienende Software.

Durch klar definierte Schnittstellen wird die Kommunikation weniger fehleranfällig. Es muss weniger interpretiert werden und Ihre Entwickler können sich endlich «auf das Wesentliche konzentrieren» – ihre Arbeit. Die Ziele und Anforderungen sind klarer definiert, die Projektsicherheit wird deutlich gesteigert. Und die Projektverantwortlichen sparen letztlich Zeit, Geld und Nerven – darum geht's!

«Viele Entwickler bleiben sehr weit von der Sichtweise des Benutzers entfernt»

Leon V. Schumacher, Gründungspartner der IT-Consulting-Firma Baxian und ehemaliger Group-CIO von Novartis, erklärt im Interview, wie man mit einem intelligenten Mock-up oder einem Prototypen die Benutzerfreundlichkeit einer Software erhöhen sowie schneller und günstiger produktiv gehen kann. Interview: Janine Aegerter

Herr Schumacher, wieso sollte Software benutzerorientiert sein?
Ist sie es nicht, ist der Nutzer unzufrieden, frustriert und gewöhnt sich nur schlecht an die Lösung. Hinzu kommt, dass Nutzer stets versuchen, sich das Leben einfacher zu machen. Es besteht daher eine grosse Wahrscheinlichkeit, dass Zusatzlösungen entstehen, um bestehende Probleme zu vereinfachen oder zu umgehen.

Ist die benutzerzentrierte Softwareentwicklung heute weit verbreitet?
Man hat erkannt, dass Usability das A und O der Softwareentwicklung ist. Das sieht man schon, wenn man den Erfolg des iPhones anschaut. Es brachte eine viel bessere Usability an den Markt und hat damit die ganze Mobilfunkindustrie auf den Kopf gestellt. Es gibt aber auch Ausnahmen, die mit einer weniger guten Usability Erfolg hatten. SAP ist ein gutes Beispiel dafür. Hier gab die integrierte Funktionalität den Ausschlag. Ich denke, solche Ausnahmen werden mit jedem Tag schwieriger, weil der User immer mehr gute Alternativen bekommt.

Ist Usability heute also wichtiger als früher?
Sie war früher genauso wichtig. Man hatte aber weniger Möglichkeiten. Heute entwickelt man eher agil, früher hat man sich am Wasserfall-Modell orientiert. Das Resultat hat folglich nicht immer den gesetzten Anforderungen entsprochen. Abgesehen davon gab es noch nicht so viele Hilfe-Tools wie heute.

Was hat sich denn im Vergleich zu früher konkret verändert?
Wenn man sich heute in der Branche umschaut, findet man immer noch traditionelle wie auch innovative Projekte. Es gibt Tools, die helfen, einen intelligenten Mock-up oder eine Visualisierung zu erstellen. Damit kann man dem User und dem Entwickler zeigen, wie das System letztlich aussehen sollte. In meiner früheren Funktion haben wir zum Beispiel auf Spezialisten wie die Ergonomen Usability AG und Tools wie iRise zurückgegriffen, um

Leon Schumacher bietet bei Baxian spezialisiertes IT-Consulting zu den Themen IT-Transformation, IT-Security und SAP an.

das Problem zu lösen. In einer sogenannten Landscape-Konsolidierung haben Entwickler umgekehrt die Möglichkeit, die Komplexität zu senken und einen voll funktionsfähigen, gestrafften Prototypen zu entwickeln, den die Nutzer dann gleich testen können.

Können Sie das an einem Beispiel veranschaulichen?
In einer früheren Funktion habe ich mit einem Team von 6 Personen in 3 Monaten die Komplexität von 20 SAP-Systemen um einen Faktor 100 senken können. Wir haben unter anderem die Anzahl Sales Order Types von 1600 auf 12 reduziert. Die Anzahl Sales Order Types ist ein guter Massstab für die Komplexität von SAP. Wir haben alle Funktionalitäten im Standard in einem voll funktionsfähigen Prototypen abgebildet und die Nutzer gebeten, diesen zu testen. Sie konnten ihr Feedback abgeben und der Prototyp wurde kurzfristig entsprechend angepasst.

Verkauft sich eine benutzerfreundliche Software besser als eine, die komplex aufgebaut ist?
Ja, ich meine schon. Solange Software die nötigen Funktionalitäten erfüllt, wird die Bedienbarkeit den Unterschied machen. Hätte es zu den Zeiten von R/3 so etwas wie iSAP von Apple mit ähnlicher Funktionalität gegeben, wäre Letzteres sicher erfolgreicher gewesen.

Worauf muss man achten, wenn man benutzerorientiert entwickeln will?
Man muss seine Nutzergruppe kennen und diese abbilden. Wer eine Software für die Stahlindustrie entwickelt, muss mit seiner Software vor allem Männer zufrieden stellen. Bei Consumer-Software hingegen ist das anders. Zudem ist es wichtig, künftige Nutzer nicht mit Bergen von Prozessabläufen auf Papier zu überfordern. Ich setze mich immer wieder für Prototypen ein, die man schnell weiterentwickelt. Damit kann man die Produktivität steigern und die Kosten beachtlich senken.

Wie muss man seine Mitarbeiter und die Projektteams führen, wenn man benutzerorientiert entwickeln will?
Erstens muss man die Schnittstellen genau definieren. Zudem muss man Entwickler und Endnutzer zusammenbringen, damit sie miteinander kommunizieren können. Hier können wieder Teams wie die Ergonomen Usability helfen, wenn das nicht intern sichergestellt werden kann. Man kann den Anforderungskatalog nicht einfach den Entwicklern über die Mauer zuwerfen und sechs Monate später ein fertiges Produkt erwarten.

Ist es für Entwickler schwierig, sich in den Nutzer hineinzuversetzen?
Nun, viele Entwickler bleiben sehr weit von der Sichtweise des Nutzers entfernt. Dies ist ein altbekanntes Problem, für das immer noch keine Lösung gefunden ist. Ideal wäre, wenn ein Entwickler sowohl das Business wie auch die IT verstünde. Solche Leute sind aber leider dünn gesät. Das liegt wahrscheinlich daran, dass sich ein Entwickler nicht unbedingt für geschäftliche Abläufe interessiert. Ein Nicht-Entwickler hingegen versteht zwar das Business, interessiert sich aber nicht unbedingt dafür, wie Code geschrieben wird und funktioniert. <