Prototyping - Theorie und Praxis

Unter dem Druck eines stetig wachsenden Softwaremarktes und komplexer werdender Anwendungen gelingt es Entwicklern heute kaum, rechtzeitig und innerhalb des geplanten Budgets qualitativ hochwertige Software zu produzieren. Besonders die Entwicklung graphischer Benutzerschnittstellen stellt hohe Anforderungen an die Beteiligten am Entwicklungsprozeß. Viele kommerzielle Projekte bringen nie fertige Produkte hervor, haben mit erheblichen Problemen zu kämpfen oder bringen zumindest nicht den ökonomischen Vorteil, der geplant war.

Erfolgsrezepte

Prototyping war schon Alfred Hitchcock´s Erfolgsrezept. Der Großmeister des Films testete Reaktionen auf Geschichten, die er bei Cocktailparties zum besten gab, experimentierte mit verschiedenen Abwandlungen der Story, die auf Kommentaren des Publikums basierten und testete schließlich Filmsequenzen am Publikum. Psycho war eines von vielen erfolgreichen Ergebnissen dieser Technik.

Auch in vielen industriellen Bereichen (Autoindustrie, Architektur etc.) ist Rapid Prototyping ein wesentlicher Bestandteil des Entwicklungsvorgangs. Es wird mindestens ein Prototyp des Endprodukts erstellt, von Benutzern getestet und, falls Verbesserungswünsche existieren, entsprechend geändert. Es wird davon ausgegangen, daß der Prototyp so schnell erstellt werden kann, daß genug Zeit bleibt, um auch grundlegende Veränderungen, die nach der Evaluation gewünscht werden, durchzuführen. Ist die Erstellung von Prototypen schnell genug, so werden mehrere Prototypingzyklen möglich, in denen der Prototyp iterativ verbessert und verfeinert wird.

Auch aus der Entwicklung interaktiver Systeme ist Rapid Prototyping mittlerweile als zentrales Element der meisten Entwicklungszyklen nicht mehr wegzudenken.

Die Erfahrungen mit konventionellen Methoden wie dem Wasserfallmodell (Anforderungsanalyse - Spezifikation - Entwurf - Implementation - Test - Wartung) waren denn auch ernüchternd. Die Gründe hiefür sind vielfältig.

Wenn keine Prototypen (in diesem Zusammenhang auch als Dialogsimulation oder ausführbare Spezifikation bezeichnet) erzeugt werden, so müssen sehr viele Designentscheidungen (komplexere Benutzerschnittstellen bestehen aus tausenden von Widgets) getroffen werden, ohne die Möglichkeit, diese visuell darzustellen. Dies erschwert die Kommunikation innerhalb des Designteams, das aus Experten verschiedener Fachbereiche (Psychologie, Ergonomie, Informatik, Visuelles Design etc.) besteht, und auch die Kommunikation zwischen Kunden, künftigen Benutzern, Managern und Entwicklern.

Benutzer und Manager können oft erst angesichts eines lauffähigen Systems die Anforderungen an ein solches System deutlich machen. Ist das Design schließlich implementiert, dann ist es zu spät, da zu aufwendig, um Änderungen vorzunehmen.

Auch Entwickler haben so ihre Probleme mit konventionellen Entwicklungszyklen. Sie lassen sich nur ungern in die Abläufe pressen, die der konventionelle Softwarelebenszyklus ihnen vorschreibt. Statt dessen vermeiden sie gerne Spezifikationen und arbeiten lieber bottom-up als top-down.

Neben Kommunikations- und methodischen Problemen existiert auch nach wie vor das Designproblem an sich. Ebensowenig wie Methoden, die automatisch zum Ziel führen, gibt es konkrete Kriterien und Richtlinien für qualitativ gute Benutzerschnittstellen an denen sich der Designer orientieren könnte. In der Literatur sind zwar eine Vielzahl psychologisch fundierter Kriterien zu finden, diese sind jedoch nicht konkret genug und können nur als grobe Richtlinien verwendet werden. Schließlich ist es der "menschliche Faktor" des Endprodukts, der entscheidet und dieser ist nach wie vor schwer zu erfassen. Es ist daher im Moment noch nicht gezielt möglich, in einem Durchgang die beste Benutzerschnittstelle für ein Problem zu finden...

Quick and dirty

Prototyping ist nur möglich, wenn der Prototyp schnell und billig erzeugt werden kann. Das Implementieren der Benutzerschnittstelle am Zielsystem, um sie danach zu evaluieren ist zu aufwendig und langwierig für Prototypingzyklen, ebenso die Anbindung zum Anwendungssystem. Je nach dem Werkzeug, das für das Prototyping verwendet wird, müssen für den Prototyp Einschränkungen in Kauf genommen werden, ein Jonglieren im Trade-off zwischen Zeit und Perfektionismus. Der Prototyp selbst sollte leicht modifizier- und erweiterbar aber nicht notwendigerweise vollständig sein.

Vordringlichster Zweck des Prototyping ist die Herstellung eines Systems, das so gut wie (mit dem verwendeten Werkzeug) möglich die Benutzerschnittstelle simuliert, sodaß der gewählte Designansatz von Designern und Testpersonen evaluiert werden kann. Je ähnlicher der Prototyp der intendierten Benutzerschnittstelle ist, desto sinnvoller ist natürlich das Ergebnis der Evaluation.

Wozu das alles?

Kommunikation ist alles

Prototyping hilft dabei, die Kommunikation zwischen den Anwendungsprogrammierern auf der einen Seite, die die Notwendigkeit von benutzerorientierter Entwicklung nicht immer verstehen, und den Designern der Benutzerschnittstelle, die ihrerseits wenig über implementationstechnische Anforderungen und Einschränkungen wissen, zu erleichtern. Das Kommunikationsproblem existiert aber auch innerhalb der Designgruppe, die idealerweise aus Fachleuten aus der Psychologie, Ergonomie, Informatik und Design besteht.

Spezifizieren geht über Implementieren

Die Verwendbarkeit von Prototypen geht weit über den ursprünglichen Zweck der Evaluation hinaus. Prototyping ist eng mit der Spezifikation von Benutzerschnittstellen verbunden. Einerseits können Spezifikations- und Repräsentationsmethoden dazu verwendet werden, um zu spezifizieren, wie der Prototyp aussehen soll, besonders, wenn die Entwicklung des Prototyps nicht von den Designern, sondern von Entwicklern oder Spezialisten vorgenommen wird. Nicht jedes Prototypingtool ist einfach bedienbar, manchmal ist auch Programmieren erforderlich.

Andererseits kann der Prototyp selbst als Spezifikation der Benutzeroberfläche angesehen werden. Wenn Designer mit Hilfe eines leicht bedienbaren Werkzeugs einen Prototypen erzeugen, dann stellt dieser die beste, weil lauffähige, Spezifikationsmöglichkeit dar, die es gibt.

Manchmal wird ein Prototyp sogar nur zum Zwecke der Spezifikation erstellt. Funktionsfähige Prototypen sind das bestmögliche Kommunikationsmittel über die Benutzerschnittstelle. Im Vergleich zu jeder Sprache, die eine Benutzerschnittstelle beschreibt, schneidet der Prototyp besser ab, was Verständlichkeit betrifft. Er ermöglicht intuitives, schnelles Erfassen der Konzepte einer Benutzerschnittstelle. Es muß für die Erklärung der Darstellungen und deren Funktionalität nicht der Umweg über eine - meist komplizierte - Spezifikationssprache gegangen werden, die vom Betrachter oder Leser erst erlernt werden muß. Prototypen können herkömmliche Spezifikationen, die dazu gedacht sind, Programmierern Information über die Benutzerschnittstelle zu vermitteln, zumindest zum Teil (abhängig von ihrer Vollständigkeit) ersetzen. Für die Entwickler hat Prototyping den Vorteil, daß sie mit Hilfe der Prototypen besser abschätzen können, welchen Aufwand die Implementation der geforderten Funktionalitäten notwendig machen wird.

Grundlagen

Im Rahmen von Grundlagenforschungen werden manchmal auch Prototypen erstellt, die nur dem Wissenserwerb dienen, ohne auf ein konkretes Zielsystem hinzuarbeiten, z.B. um neue Konzepte für die Gestaltung von Benutzerschnittstellen evaluieren zu können. Für Benutzertests ist ein Prototyp der neuen Technik unbedingt notwendig.

Prototypingtechniken

Softwareprototypen können verschiedene Formen annehmen und unterscheiden sich je nach verwendeter Technik. Zu den grundlegenden Techniken zählen Slide Shows, Wizard of Oz und vollständige Prototypen.

Slide Show

Eine Slide Show ist eine Menge von Bildschirmen, die dem Benutzer in einer bestimmten Reihenfolge gezeigt werden. Dadurch wird auf eine simple Art und Weise ein Ablauf in der Benutzerschnittstelle simuliert. Einfache Slideshowprototypen können auch mit Papier und Bleistift, Graphikprogrammen, Präsentationsprogrammen oder Hypermedien erzeugt werden. Trotz der Einfachheit dieser Technik ermöglicht sie die Verwendung von Graphiken und groben Abläufen und kann daher sehr gut zu Kommunikationszwecken genutzt werden.

Wizard of Oz

Aus Märchen kann man noch etwas lernen. Bei dieser Technik simuliert ein Mensch den Computer. Einsatzmöglichkeiten ergeben sich z.B. wenn Prototyping mangels geeigneter Tools nicht möglich ist. Für Frage-und-Antwort Dialoge und natürlichsprachliche Ein- und Ausgabe ist diese Technik sehr gut geeignet und vor allem sehr kostengünstig.

Vollständige Prototypen

Vollständig animierte Prototypen sind sehr viel schwieriger zu erstellen. Die Zeit und Ressourcen, die benötigt werden hängen von der Komplexität des Dialogs, den Graphiken und der Effizienz des verwendeten Prototypers ab. Mit Hilfe dieser Technik kann eine detailliertere Evaluation der Benutzerschnittstelle durchgeführt werden, sofern dies notwendig ist.

Horizontal und Vertikal

Ein Prototyp muß nicht bzw. soll nicht unbedingt ein perfektes Artefakt sein. Auch Papier-und-Bleistift Skizzen von Bildschirmentwürfen stellen einen einfachen Prototyp dar, der evaluiert werden kann. Die Darstellung auf Papier genügt z.B., um die Erkennbarkeit von Ikonen zu testen. Bezüglich der angestrebten Vollständigkeit unterscheidet man horizontales und vertikales Prototyping. Beim vertikalen Prototyping werden einige Teile des künftigen Systems bis ins Detail simuliert. Es kann eine realistische Evaluation, aber eben nur für einige Teile erfolgen. Beim horizontalen Prototyping wird ein möglichst breiter Prototyp erzeugt, der einen Überblick über das gesamte System bieten soll, wobei ev. auf Details verzichtet wird. Vertikales Prototyping sollte für Grundlagenforschungen und Tests neu einzusetzender Interaktionstechniken verwendet werden. Für Manager wiederum wird ein Überblick über das Gesamtsystem interessanter sein, ebenso für Entwickler, die den Implementationsaufwand für das Gesamtsystem abschätzen sollen.

Als Szenario wird ein Prototyp bezeichnet, der in beide Richtungen nur eingeschränkt funktioniert, dafür aber billig und schnell erzeugbar ist, z.B. ein Papier-und-Bleistift Prototyp.

Explorativ oder experimentell

Je nach den Zielen, die beim Prototyping verfolgt werden, können exploratives und experimentelles Prototyping unterschieden werden. Exploratives Prototyping wird in der Anforderungsanalyse angewandt, um die Anforderungen von Benutzern und Managern an das künftige System zu klären. Die Anforderungen an den Prototyp selbst sind dabei noch sehr unklar und lassen viele (Design) Möglichkeiten offen. Beim experimentellen Prototyping stehen technische Fragen im Vordergrund. Es soll Entwicklern helfen, die Machbarkeit eines Systems einzuschätzen und Lösungsvorschläge zu erarbeiten.

Revolutionär versus evolutionär

Das wichtigste Klassifikationskriterium für Prototyping ist die Unterscheidung in revolutionäres und evolutionäres Prototyping.

Beim revolutionären Prototyping werden möglichst früh Prototypen erstellt, evaluiert und danach verworfen. Das Zielsystem wird unabhängig davon, aber basierend auf den Erkenntnissen aus dem Prototyping, entwickelt. Beim evolutionären Prototyping wird der Prototyp immer weiter verbessert und schließlich als Zielsystem verwendet. Der Unterschied zwischen Prototyp und Endprodukt ist dann nicht mehr deutlich, die Einsatzmöglichkeit des Prototyps erstreckt sich in diesem Fall von frühen Skizzen und ersten Evaluationen bis hin zur Entwicklung. Dies kann z.B. der Fall sein, wenn der Prototyp ohnehin in einer konventionellen Programmiersprache codiert werden muß, weil für die gewählte Plattform kein geeignetes Prototypingtool zur Verfügung steht. Wird der Prototyp als geeignet angesehen, so müssen nur noch Programmverfeinerungen durchgeführt und die Anwendungsfunktionalität hinzugefügt werden. In vielen Fällen wird es auch wünschenswert sein, schon vor Fertigstellung des Prototypen bzw. der Benutzerschnittstelle Anwendungsfunktionalität hinzuzufügen, um ein realistischeres Testen des Prototypen zu ermöglichen.

Darwin...

Revolutionäres Prototyping ist nicht immer leicht handzuhaben. Jeder der Beteiligten hat viel Arbeit in einen Prototyp investiert, der nun nicht mehr verwendet wird. Besonders für die Erzeuger von Prototypen ist es deprimierend, immer wieder Software herzustellen, die nie von Benutzern verwendet wird.

Alle Beteiligten am Entwicklungsprozeß erwarten, daß das Endprodukt genauso aussieht, wie der Prototyp, was aber nicht immer der Fall sein kann. Oft bedingen Einschränkungen des Zielsystems oder Zeitmangel, daß das Endsystem sich vom Prototyp unterscheidet. Z.B. wird oft auf fortschrittliche Designansätze verzichtet, da von der gewählten Entwicklungsumgebung nur Standards effizient unterstützt werden.

Auf den ersten Blick erscheint evolutionäres Prototyping daher als vorteilhafter, da der Prototyp und damit Ressourcen nicht vernichtet, sondern wiederverwendet werden. Aus diesem Grund tendieren Manager vielfach dazu, evolutionäres Prototyping zu bevorzugen. Aber nicht alles, was glänzt, ist auch gleich ideal.

oder Wegwerfgesellschaft?

Leider scheitert der gutgemeinte Recyclinggedanke häufig am Mangel geeigneter Prototypingwerkzeuge. Je einfacher ein Prototyp mit dem Werkzeug zu erstellen ist, desto komplexer ist häufig der Code, der durch ein Werkzeug erzeugt wird und desto langsamer ist der Prototyp. Prototyper interpretieren häufig Code, anstatt ihn zu übersetzen. Dies hat den Vorteil, daß der Prototyp schneller zur Verfügung steht, macht ihn aber ebenfalls langsamer. Unterschiedliche Performance zwischen Endsystem und Prototyp kann sich bei Benutzertests als sehr nachteilig erweisen, da Performance auch ein wichtiges Usability-Kriterium ist.

Weiters wird von Prototypern verlangt, daß Datentypfehler, syntaktische und semantische Codierfehler etc. nicht in langen Fehlerlisten und noch längeren Zeiten des Ausbesserns enden, wie dies bei Programmiersprachen normalerweise der Fall ist. Statt dessen sollte der Prototyp möglichst in jedem Stadium lauffähig sein, um lange Pausen zu vermeiden. Auch diese Einschränkungen verlangsamen den Prototyp und ermöglichen Fehler zur Laufzeit. Um eine gute Performance des Zielsystems zu erreichen, begibt man sich daher oft auf eine niedrigere Programmiersprachenebene. Oft stehen am Zielsystem keine geeigneten Prototypingwerkzeuge zur Verfügung. Der Prototyp wird dann auf einer alternativen Plattform, getrennt vom Zielsystem entwickelt. Unter Umständen wird dadurch die Verwendbarkeit des Prototypen für die Evaluation des Designs eingeschränkt, wenn z.B. nicht die gleichen Eingabegeräte vorhanden sind wie am Zielsystem.

Wenn das Prototyping Benutzern und Managern dabei helfen soll, Anforderungen zu definieren und schnell einige Alternativen durchzutesten, dann ist revolutionäres Prototyping besser geeignet. Es kann dabei helfen, Anforderungen zu validieren, die Vollständigkeit der Anforderungen zu überprüfen und dient als Spezifikation des Designs. Wenn diese Abschnitte im Entwicklungsprozeß schnell und kostengünstig durchgeführt werden können, dann trägt revolutionäres Prototyping zur Vereinfachung des gesamten Entwicklungsprozesses bei.

Risiken

Neben den Vorteilen, die Prototyping aufweist, entstehen aber auch einige Schwierigkeiten bei der Entwicklung, die allerdings durch Anwendung großer Sorgfalt vermieden werden können. Zunächst muß sichergestellt sein, daß alle Beteiligten mit der Vorgangsweise des Prototyping einverstanden sind. Die Möglichkeit, einen Prototyp zu verbessern hängt vom Willen und Können der künftigen Benutzer ab, entsprechende konstruktive Kritik zu liefern. Systementwickler nehmen die Entwicklung von Prototypen oft nicht wirklich ernst und neigen zu Ungenauigkeiten mit dem Argument, daß sie kein "richtiges" Produkt produzieren. Manager sehen Prototyping z.T. als Verschwendung von Ressourcen. Die Motivation aller Beteiligten ist eine Grundvoraussetzung für einen erfolgreichen Prototypingprozeß.
Bei großen Systemen kann auch der Prototyp sehr groß werden. Wird er dann noch immer als eine Art Spielzeug angesehen, kann dies zu Problemen beim Prototyping führen. Geeignete Methoden und Werkzeuge können hier Abhilfe schaffen. Ernsthafte Probleme können auch auftreten, wenn die Beteiligten den Prototyp als Endprodukt ansehen. Die Begeisterung von Benutzern und Entwicklern an weiterem Prototyping kann schwinden, wenn sie einen lauffähigen Prototypen sehen. Aber auch das Gegenteil kann, besonders beim revolutionären Prototyping ein Problem, eintreten. Entwickler möchten einen Prototypen immer weiter verbessern. Um den Prototypingprozeß nicht endlos werden zu lassen, muß hier als Ziel gesetzt werden es "gut genug" zu machen, eine heikle Managementaufgabe.

Chancen

Prototyping ist ein komplexer Vorgang, der hohe Anforderungen an alle Beteiligten, besonders an das Management stellt. Als Lohn der Anstrengungen steht ein verkürzter Entwicklungsprozeß und niedrigere Kosten im Vergleich zu konventionellen Vorgangsweisen, sowie als Ergebnis qualitativ bessere Benutzerschnittstellen, die die Konkurrenzfähigkeit sichern können, in Aussicht.

Referenzen

Arthur L. J.: Rapid Evolutionary Development: Requirements, Prototyping & Software Creation, Wiley, New York, Toronto, 1992

Bass L., Coutaz J.: Developing Software for the User Interface, Addison Wesley, Reading, MA, 1991

Berghel H.: New Wave Prototyping: Use and Abuse of Vacuous Prototypes, Interactions, Bd. 1, Nr. 2, April 1994, S. 49 - 54

Bösze J., Aschacher D.: Prototyping mit "Hyper-Tools" in: Ackermann D., Ulich E. (Ed.): Software Ergonomie '91, Teubner, Stuttgart, S. 119 - 129

Carroll J. M., Rosson M. B.: Usability Specifications as a Tool in interactive Development, in: Hartson R. H. (Ed.): Advances in Human Computer Interaction, Bd. 1, Ablex, Norwood, NJ, 1985, S. 1 - 28

Downton A.: Engineering the Human-Computer Interface, Student Edition, McGraw Hill, London, 1993

Felix D., Krueger H.: Iterative Prototyping of User Interfaces, in: Smith M. J., Salvendy G. (Ed.) : Advances in Human Factors/Ergonomics, Human-Computer Interaction: Applications and Case Studies, S. 158 - 162

Floyd D.: A Systematic Look at Prototyping in: Budde R., Kuhlenkamp K., Züllighoven H.: Prototyping - An Approach to Evolutionary System Development, Springer, Berlin, 1984, S. 1 - 19

Floyd C., Mehl W.-M., Reisin F.-M., Schmidt G., Wolf G.: Out of Scandinavia: Alternative approaches to software design and system development, Human Computer Interaction, Nr. 4, 1989, S. 252 - 350

Hartson H. R., Hix D.: Human-computer interface development: concepts and systems for its management, ACM Computing Surveys, Bd. 21, Nr. 1, März 1989, S. 6 - 92

Hartson H. R., Smith E. C.: Rapid prototyping in human-computer interface development, Interacting with Computers, Bd. 3, Nr. 1, S. 51 - 91, 1991

Hershmann D.: The Effects of Practical Business Constraints on User Interface Design, Design Briefings: Redesigning Existing Products, Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems, 1995, S. 531-537

Hix D., Hartson H. R.: Developing User Interfaces, Wiley, New York, Toronto, 1993

Kieback A., Lichter H., Schneider-Hufschmidt M., Züllighoven H.: Prototyping in industriellen Software-Projekten, Informatik-Spektrum, Nr. 15, 1992, S. 65 - 77

Overmyer S. P.: Revolutionary vs. Evolutionary Rapid Prototyping: Balancing Software Productivity and HCI Design Concerns, in: Bullinger H.-J.: Human Aspects in Computing: Design and Use of Interactive Systems and Work with Terminals, Elsevier Science Publishers, 1991, S. 303 - 307

Pomberger G., Bischofberger W., Kolb D., Pree W., Schlemm H.: Prototyping-Oriented Software Development - Concepts and Tools, Structured Programming, Nr. 12, 1991, S. 43 - 60

Wilson J., Rosenberg D.: Rapid Prototyping for User Interface Design, in: Helander M.: Handbook of Human-Computer Interaction, North Holland, 1988, S. 859 - 875