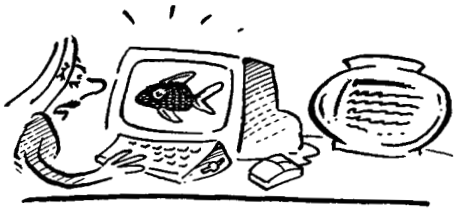
Ein unbekannter und dann auch noch schön komplizierter Fachausdruck macht sich doch immer gut als Überschrift – man stutzt, man guckt, und dann will man wissen, was das eigentlich ist – Usability Engineering. ›Engineering‹ (eigentlich: Ingenieurswesen) bedeutet hier Softwareentwicklung, und ›Usability‹ leitet sich ab von Gebrauch, Benutzung. Beides zusammen heißt also, sagen wir mal: ›benutzungsorientierte Softwareentwicklung‹. Müßte ja eigentlich eine Selbstverständlichkeit sein (für wen oder was sollte Software denn sonst entwickelt werden), ist es aber nicht. Matthias Rauterberg stellt bewährte, aber leider nicht allgemein bekannte Methoden vor, wie alle Beteiligten durch Kommunikation und Kooperation zu besserer Software kommen könnten.

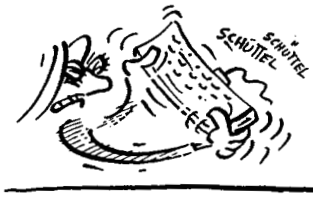
Usability Engineering Now!

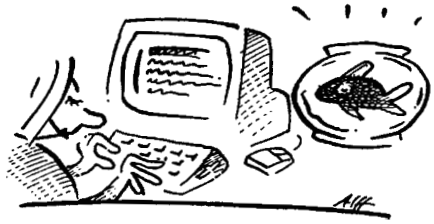
Der Computereinsatz direkt am Arbeitsplatz bedeutet nicht nur eine einschneidende Veränderung der Arbeitsbedingungen vieler Angestellter, sondern auch die Gefahr, daß die Benutzung und Beherrschung des Computers zu einer zusätzlichen, mühsamen Aufgabe wird, obwohl das System eigentlich als Unterstützung bei den eigentlichen Aufgaben gedacht war – etwa um den arbeitenden Menschen von unzumutbaren, ständig wiederholten Routinetätigkeiten zu entlasten.

Verantwortlich dafür sind eine ganze Reihe von Problemen und Schwachstellen bei der üblichen Softwareentwicklung. Vor allem stellt es sich immer wieder als schwierig heraus, ein gemeinsames Verständnis aller betroffenen Personengruppen über den zu automatisierenden Arbeitsanteil herzustellen, also ***gemeinsam*** verbindliche Antworten auf die Frage zu finden, ***ob***, ***wo*** und ***wie*** die neue Technik eingesetzt werden soll.

Dafür ist es vor allem nötig, alle Eigenschaften des neu zu gestaltenden Arbeitssystems und der neuen Arbeitsaufgaben genau zu bestimmen. Und das wiederum erfordert es, das Wissen der BenutzerInnen, der Arbeits- und OrganisationsgestalterInnen sowie der InformatikerInnen zusammenzuführen. Die BenutzerInnen müssen dabei nicht nur ihr Wissen über die Bearbeitung von Aufgaben beisteuern, sie müssen auch Wissen über die Handhabung des neuen Systems erwerben. Das alles zusammen läßt sich selbstverständlich am besten durch ein gemeinsames Vorgehen aller Beteiligten, einschließlich des Auftraggebers, erreichen.







Heute lautet die Frage also nicht mehr, ***ob*** BenutzerInnen überhaupt beteiligt werden sollen, sondern ***wie*** ihr Fachwissen zur Entwicklung guter, aufgaben- und benutzungsorientierter Software genutzt werden kann!

Eine ganze Reihe von Lösungsmöglichkeiten ist bereits in der Praxis der Softwareentwicklung entstanden, wobei darüber Einigkeit herrscht, wie vorteilhaft es ist, alle betroffenen Gruppen einzubeziehen. Allerdings werden auch Barrieren deutlich – an erster Stelle eine Kommunikationsbarriere …

Benutzer: »Also, was ich will, ist, Einzelheiten zu neuen Büchern eingeben, um diese dann später …«

Entwickler: »Ja, ist klar. Wir müssen deshalb zunächst Schlüsselmerkmale für Selektionen definieren.«

B: »Ääh … ich meine thematisch … ja, und katalogisieren, Schlüssel ändern …«

E: »Was nicht so einfach ist bei dieser DB …«

B: »Ach? Und dann will ich nach Autoren und Jahrgang sortieren, auswählen und verschiedene Listen erstellen und ausdrucken.«

E: »Gut, dann machen wir ein Menü ›Erfassung/Mutation/Anzeigen‹, sauber getrennt, damit bei ›Anzeigen‹ auch wirklich nichts geändert werden kann.«

B: »Aber ich muß da doch was ändern können …«

E: »Ja, ja. Dann ein Menü ›Selektion‹, wo Sie auch nach Autor oder Jahrgang sortieren können."

B: »Nicht ›oder‹, und Jahrgang – nach beidem!«

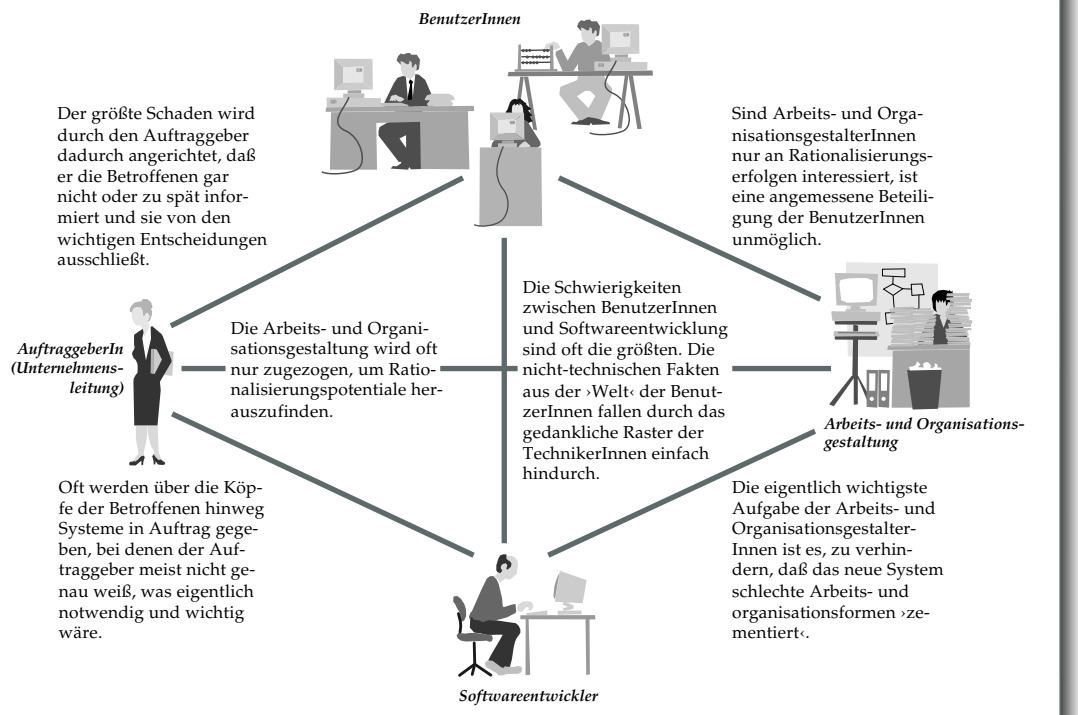
E: »Tja, das geht bei dieser DB nicht! Wieviele Records haben Sie denn überhaupt?«

B: »Records ….???«

Kommunikationsprobleme dieser Art tauchen – siehe Abbildung auf Seite 58 – mehr oder weniger schwerwiegend zwischen allen Beteiligten auf. Sie zu überwinden, ist Hauptziel der Benutzerbeteiligung. Die dabei jetzt schon am häufigsten eingesetzten Me-

thoden allerdings – Workshops, Review-Sitzungen, ›User Groups‹, betriebsinterne Umfragen, Alpha- oder Beta-Tests, Kontaktaufnahme zu speziellen Kunden oder BenutzerInnen, ›Hot Line‹ – dienen fast ausschließlich der Überprüfung der Software auf ihre funktionale Vollständigkeit und Korrektheit, und müssen deshalb hier nicht näher erläutert werden. Benutzungsprobleme im handlungs- und arbeitspsychologischen Sinne hingegen werden noch kaum berücksichtigt – und hier setzt die Methode der benutzungsorientierten, der ***›Usability-Tests‹*** an. Sie sind deshalb nötig, weil bestimmte Softwareeigenschaften eben nur im konkreten, praxisähnlichen Einsatz meßbar sind!

Kommunikationswege und -barrieren zwischen den am Prozeß der Softwareentwicklung beteiligten Personengruppen



Normen allein genügen nicht

Auch Normen, Kriterien und Richtlinien sind in diesem Prozeß selbstverständlich notwendig und nützlich, tragen sie doch dazu bei, Softwaregestaltung zu vereinheitlichen, Entscheidungen nicht immer wieder neu treffen zu müssen, und detailliertere Anforderungen formulieren zu können. Allein aber reichen sie bei weitem nicht aus, um benutzungsgerechte Systeme zu entwickeln.

Also müssen alle Betroffenen in einem ausreichenden Maße beteiligt werden. Dies erfolgt, wenn es erfolgt, in der Regel durch die Bildung von Projektteams und Arbeitsgruppen.

Daß Teamarbeit die geeignetste Organisationsform ist, um (Software-)Projekte abzuwickeln, ist inzwischen unbestritten. Der Erfolg allerdings hängt ab von einer ganzen Reihe von Kriterien wie beispielsweise Fachkompetenz, Entscheidungskompetenzen, Beteiligungsmöglichkeiten, Motivation, Kontinuität oder auch Fluktuation. Dabei führt eine (zu) starke Funktions- und Arbeitsteilung in und zwischen Projektteams zu einem Mangel an Transparenz, an Eigeninitiative und an der Bereitschaft, Verantwortung zu übernehmen. Auch die Lern- und Entwicklungsmöglichkeiten bei der Arbeit sind eingeschränkt, oft mit der Folge geringer Motivation und Produktivität.

Deshalb arbeiten Projektteams dann besonders effizient, wenn sie erstens weitgehend selbständig und relativ unabhängig arbeiten können, wenn zweitens das Projekt einen inneren Aufgabenzusammenhang bietet, und wenn drittens das Projektteam sich mit dem zu entwickelnden Produkt zu identifizieren vermag. Konkret: Einer Gruppe von AnalytikerInnen/ProgrammiererInnen, ergänzt durch beispielsweise BenutzervertreterInnen, ist eine ganzheitliche Projektaufgabe – die möglichst viele miteinander zusammenhängende Teilaufgaben ›von A bis Z‹ umfaßt – zu übertragen.

Für eine wirksame Beteiligung der BenutzerInnen gibt es dabei verschiedene Ansätze:

So können die BenutzerInnen (nur) zu definierten Zeitpunkten – etwa beim Vorliegen eines (Zwischen-) Ergebnisses – einbezogen werden, um Vorschläge, Prototypen oder System-Module zu beurteilen. Hier einige Beispiele für diese eher passive Form der Beteiligung:

1. Die Vorschläge werden den BenutzerInnen schriftlich vorgelegt; mit einem Fragebogen wird eine Beurteilung nach bestimmten Merkmalen erfragt und

Verbesserungsvorschläge werden eingeholt (›Zeichnen Sie bitte selbst die von Ihnen gewünschten Änderungen ein!‹); häufig werden auch Alternativen angeboten (›Welche Maske – A oder B – finden Sie besser?‹).
2. In einem speziell einberufenen ›Workshop‹ oder auch auf einem Abteilungstreffen stellen die EntwicklerInnen ihre Vorschläge vor und diskutieren sie mit den BenutzerInnen.
3. Bildschirmmasken werden in einem Test mit einem Teil der BenutzerInnen anhand realistischer Aufgaben geprüft; dabei werden objektive Daten erhoben (Wie schnell findet ein Benutzer die Information X auf Maske Y – eventuell noch im Vergleich mit Maske Z?) und subjektive Daten (persönliche Einschätzung).

Durchgehende Beteiligung bringt aktive, kontinuierliche Mitarbeit

Eine weitere Möglichkeit ist die durchgehende ***›prozessuale‹*** Beteiligung der BenutzerInnen (oder auch ihrer Vertretung), bei der sie zu einem mehr oder weniger großen Teil ihrer Arbeitszeit kontinuierlich, also vom Anfang bis zum Ende, im Projekt mitarbeiten und dabei Teilaufgaben übernehmen. Dabei sind die BenutzerInnen nicht darauf beschränkt, Stellungnahmen zu Vorgegebenem abzugeben, sondern sie können selbst oder in Zusammenarbeit mit den EntwicklerInnen Vorschläge entwerfen und dabei ihre Erfahrung konstruktiv umsetzen. Hier liegt also das Schwergewicht auf aktiver, kontinuierlicher Mitarbeit bei der Entwicklung – zwei Beispiele:

1. Zwei BenutzerInnen arbeiten zu 50 Prozent ihrer Arbeitszeit in der Projektgruppe. Sie wirken aktiv mit bei der Erarbeitung von Konzepten zum Funktionsumfang, zum Informationsgehalt und zur Benutzungsoberfläche; außerdem ist es ihre Aufgabe, den Kontakt zu den übrigen 40 Benutzern sicherzustellen (Weitergabe von Informationen, Einholen von Rückmeldungen, Organisation und Durchführung von Benutzertreffen).
2. Vier BenutzerInnen und ein Programmierer arbeiten in einer speziellen Kleingruppe, die von der Projektgruppe den Auftrag bekommen hat, spezielle Masken, Formulare und Listen zu entwerfen und zu testen. Eine andere Gruppe mit zwei ProgrammiererInnen und einem Benutzer entwirft ein Konzept für die Benutzungsoberfläche und erstellt Prototypen.

Die bisherigen Beispiele zeigen bereits, daß es um eine weitere Frage geht: Sollen die BenutzerInnen ***direkt*** (beispielsweise alle betroffenen BenutzerInnen in einem Workshop mit EntwicklerInnen) oder nur ***indirekt*** (über eine Benutzervertretung, ein Koordinationsgremium, den Betriebsrat usw.) in die Entwicklung einbezogen werden. Dabei geht es vor allem auch um den Weg, den Informationen vom Benutzer bis zum Entwickler zurücklegen müssen. Der direkte Einbezug eröffnet Chancen zum 1:1-Austausch von Informationen, ist aber bei großer BenutzerInnenzahl schwierig zu realisieren (Entwickler: ›Ich kann doch nicht mit allen 150 Leuten sprechen!‹). Aus zeitlichen, organisatorischen und ökonomischen Gründen bietet sich deshalb häufig die indirekte Form an. Dabei besteht allerdings, wie bei allen Informationsflüssen, die über mehrere Stellen verlaufen, die Gefahr, daß die ursprünglichen Informationen gefiltert und verzerrt am Bestimmungsort ankommen. Beispiele:

1. Drei ***BenutzervertreterInnen*** sammelten die Meinungen ihrer ArbeitskollegInnen zum vorgeschlagenen Maskendesign und übergaben sie einem ***Koordinator***. Der bereitete die 150 schriftlichen Beurteilungen auf, verdichtete sie, ließ das (seiner Meinung nach) Unwichtige weg und ergänzte ›Fehlendes‹. Als Einleitung fügte er dann seine Interpretation des Haupttrends bei, und übergab das alles dem ***Projektleiter***. Dieser las vor allem die Einleitung, überflog noch einige Abschnitte des Berichts und übermittelte dann – befriedigt vom Gesamtergebnis – den ***Programmierern*** einzelne Änderungshinweise. Große Verwunderung, als sich hinterher die BenutzerInnen darüber beschwerten, daß die ihrer Meinung nach wichtigsten Vorschläge überhaupt nicht berücksichtigt worden waren!
2. Ein ***Projektleiter*** und ein ***Entwickler*** verbrachten drei Monate mit der Aufarbeitung von 367 ***Benutzeranforderungen*** (inklusive Rückfragen); wichtige widersprüchliche Anforderungen wurden direkt in einem Workshop geklärt …

Wichtig: Bei jeder Form indirekter Beteiligung bestimmt die Auswahl der Benutzervertretung das Ergebnis des Prozesses maßgeblich. Werden unmotivierte, schlecht qualifizierte (beispielsweise zur Zeit in der Fachabteilung gerade abkömmliche) BenutzerInnen beigezogen, sind Probleme vorprogrammiert! Deshalb ist – neben der Repräsentativität – sorgfältig auf die Motivation, Qualifikation sowie soziale und kommunikative Fertigkeiten der BenutzervertreterInnen zu achten!

Softwareergonomie beurteilen mit Checklisten

Checklisten für die Bewertung der Benutzungsfreundlichkeit von Softwareprodukten werden häufig und gerne eingesetzt. Checklisten weisen einige Stärken auf, denn sie …

… verringern den Bewertungsaufwand erheblich; deshalb ist auch das wirtschaftliche Interesse an diesem Verfahren groß;
… helfen zu verhindern, daß wichtige Aspekte bei der Bewertung vergessen werden;

Deutschsprachige Checklisten sind zu finden in:

C. Baitsch u.a., Computerunterstützte Büroarbeit, Verlag der Fachvereine, 1989;

EDV im Büro – Handbuch zur menschengerechten Gestaltung, Oldenbourg-Verlag 1990;

Siemens Nixdorf Informationssysteme, Alpha-Styleguide-Checkliste 1.0, München 1992, Bestell-Nr. U8557-J-Z147-1

… eignen sich besonders für die Bearbeitung von Routinefällen;
… enthalten die bedeutsamen Aspekte in komprimierter Form;

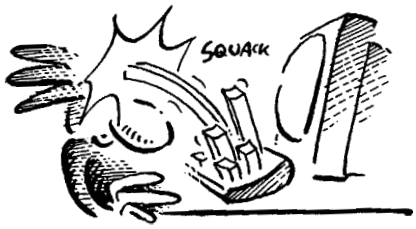
Diesen Vorteilen stehen aber auch einige Schwächen gegenüber – Checklisten …

… können Sachverstand nicht ersetzen; das Arbeiten mit ihnen erfordert hinreichende Kenntnis von Aufgaben, Problemen, Zusammenhängen und Lösungsansätzen;
… sind weniger zuverlässig als anderen Verfahren, da bewußtes oder unbewußtes Mißverstehen der Fragen möglich ist;
… bieten nur begrenzte Möglichkeiten zur eingehenden Erläuterung der einzelnen Fragen;
… lassen spezielle Aspekte, die nicht in das Raster der Checkliste hineinpassen, unberücksichtigt.

Verschiedene Typen von Benutzertests (›Usability Tests‹)

Bei den Benutzertests werden drei Methoden unterschieden und hier vorgestellt – die ›aufgabenorientierten‹, die ›induktiven‹ und die ›deduktiven‹ Benutzertests.

Aufgabenorientierte Benutzertests dienen dazu, Vorschläge für eine benutzerangemessene Gestaltung von Arbeitsabläufen zu gewinnen. Dafür wird eine Arbeitssituation möglichst realistisch nachgestellt. Aufgabenorientierte Benutzertests lassen sich beispielsweise auch in Workshops mit BenutzerInnen und EntwicklerInnen einsetzen, um eine typische Arbeitssituation im Rollenspiel nachzuspielen und so den EntwicklerInnen eine direkte und anschauliche Rückmeldung über die Situation zu geben. Ein Benutzer spielt dabei eine typische oder geplante Arbeitssituation mit entsprechend abgesprochenen Aufgaben durch. Dabei wird – wenn vorhanden – ein Prototyp der Hard-/Software benutzt. Ist kein richtiger Prototyp da, kann er durchaus auch mit einfachen Hilfsmitteln (Pappschachtel als Computerattrappe und Folien oder Papierblätter als Maskenersatz) simuliert werden. Neben dem Testleiter beobachten und kommentieren die anderen BenutzerInnen die Situation, beispielsweise wenn sie Unterschiede zu der von ihnen bevorzugten Bearbeitungsweise feststellen. Am besten läßt sich ein solcher aufgabenorientierter Benutzertests direkt am Arbeitsplatz oder in unmittelbarer Nähe dazu durchführen.



›Induktive‹ (Duden: ›vom Einzelnen zum Allgemeinen hinführend‹) Benutzertests lassen sich immer dann einsetzen, wenn nur ***eine*** Version der zu testenden Software vorliegt. ›Deduktive‹ (Duden: ›den Einzelfall aus dem Allgemeinen ableitend‹) Benutzertests hingegen dienen dem Zweck, zwischen mehreren Systemalternativen zu entscheiden – erst in zweiter Linie geht es darum, Vorschläge zur Gestaltung und Verbesserung zu gewinnen. In beiden Fällen ist eine Anzahl von Bedingungen zu beachten:

Damit keine unnötigen Fehler auftreten können (etwa durch unzureichende Funktionalität des Systems), sollte das zu testende System schon soweit entwickelt sein, daß es sich möglichst realitätsgerecht verhält. Diese Anforderung ist bei deduktiven, also vergleichenden Tests besonders hoch anzusetzen, da eine Entscheidung zwischen zwei oder mehr Systemen nur dann begründet getroffen werden kann, wenn Funktionen, Arbeitsgeschwindigkeit und ähnliches wirklich miteinander verglichen werden können.

Alsdann sind eine oder mehrere Aufgaben festzulegen, die auf die zu testenden Systemteile abgestimmt sind. Diese Aufgaben sind dem typischen Aufgabenbereich der zukünftigen BenutzerInnen zu entnehmen. Ist der Aufgabenbereich nicht oder nur vage bekannt (das wird bei neuen Systemen häufiger der Fall sein), sollten zumindest typische Teilaufgaben definiert sein. Die einzelne, zu testende Aufgabe sollte nicht zu umfassend, aber auch nicht zu einfach sein – typische Bearbeitungszeiten liegen zwischen fünf und fünfzehn Minuten.

Die Formulierung der Aufgaben sollte den BenutzerInnen während der Aufgabenbearbeitung schriftlich vorliegen. Die Aufgabenbeschreibung hat das unterschiedliche fachliche Vorwissen der Benutzer zu berücksichtigen; bei hohem fachspezifischem Vorwissen ist die Beschreibung möglichst ***problemorientiert*** abzufassen, bei geringem fachspezifischen Vorwissen ist die Beschreibung ***handlungsorientiert*** zu halten …

Problemorientierte Aufgabenbeschreibung: »Erstellen Sie bitte einen Brief mit folgendem Inhalt: … und bereiten Sie ihn zum Eintüten und Versenden an folgende Adressen: … vor. Benutzen Sie dabei bitte als Briefvorlage das Dokument mit dem Namen: …«

Handlungsorientierte Aufgabenbeschreibung: »Laden Sie bitte das Textdokument mit folgendem Namen: … und ergänzen Sie es um den folgenden Inhalt: ... Versehen Sie dieses Textdokument mit allen für die Serienbrieferstellung notwendigen Steueranweisungen. …«

Die handlungsorientierte Aufgabenbeschreibung soll verhindern helfen, daß die beobachteten Benutzungsprobleme vielleicht überwiegend durch fehlen-

des Fachwissen (hier: Korrespondenzwesen) zustande gekommen sind.

Im Unterschied zum aufgabenorientierten Benutzertest sind beim induktiven und beim deduktiven Benutzertest ***mehrere*** Benutzer beteiligt. Die Testgruppe sollte möglichst ***repräsentativ*** für die Zusammensetzung der späteren BenutzerInnen sein und mindestens sechs Testpersonen umfassen. Die ausgewählten Testpersonen sollten das zu testende System ***nicht*** kennen. Weil in dem für einen Test in Frage kommenden Personenkreis diese ›Eigenschaft‹ bei längeren Entwicklungsphasen oder im Falle von Weiterentwicklungen immer seltener wird, muß die Vorerfahrung mit dem System oder mit ähnlichen Systemen mit erhoben und bei der Auswertung dann berücksichtigt werden.

Die Testumgebung

Anzustreben ist eine den realen Einsatzbedingungen möglichst entsprechende Testumgebung. Da es für die spätere Auswertung sehr wichtig ist, das Verhalten der einzelnen Benutzer und das entsprechende Systemverhalten aufzuzeichnen (mit Video, Tonband und ›Screen Recording‹, einer Technik, mit der alle Bildschirmanzeigen und -aktionen durch ein spezielles Programm aufgezeichnet und später ›abgespielt‹ werden können), empfiehlt es sich, einen ruhigen, abgeschlossenen Raum zu benutzen.

Der Testleiter sollte sich stets um die Schaffung eines vertrauensvollen und für Kritik offenen Klimas bemühen. Dafür sind die folgenden drei Bedingungen besonders zu berücksichtigen:

1. Mit möglichst allgemeinverständlichen Worten wird dem Benutzer Ziel und Zweck des Tests erläutert (›Es handelt sich um den Test eines Prototypen in einem frühen Entwicklungsstadium …‹).
2. Besonders wichtig ist es, dem Benutzer verständlich zu machen, daß ***das System*** und ***nicht*** er als Benutzer getestet werden soll. Im Gegenteil: Jede Art von Schwierigkeiten die der Benutzer bei der Aufgabenbearbeitung feststellt, sind von besonderem Interesse!
3. Jedem Benutzer muß zugesichert werden, daß er die Durchführung des Tests jederzeit unterbrechen und sogar abbrechen kann, ohne daß irgendwelche negativen Konsequenzen (etwa der Wegfall einer zugesagten Bezahlung) zu befürchten sind. Die vollständige Freiwilligkeit der Teilnahme und Zusicherung der Vertraulichkeit ist unabdingbar für die Gewinnung von brauchbaren Testergebnissen.

Um während der Aufgabenbearbeitung erfassen zu können, was jeweils die eigentlichen Absichten und Handlungsziele sind, werden die Testpersonen gebeten ›laut zu denken‹, was vielen allerdings sehr schwer fällt. Es empfiehlt sich daher, dem Benutzer zu verdeutlichen, was mit ›lautem Denken‹ gemeint ist, und mit ihm einige Übungsbeispiele gemeinsam durchzugehen.

Gemessen oder festgehalten werden ***qualitative*** Meßgrößen (problematische Benutzungssituationen, Fehlerarten …) und ***quantitative*** Meßgrößen (Bearbeitungsdauer, Anlernzeit, Überlegungszeit, Fehleranzahl …).

Der Testleiter hält sich während der Aufgabenbearbeitung ruhig im Hintergrund. Für ihn ist es besonders wichtig, den Wunsch zurückzuhalten, dem Benutzer in problematischen Situationen sofort zu helfen. Ein zu frühes Eingreifen des Testleiters hindert den Benutzer, vielleicht eine eigene Lösung zu finden. Als Testleiter sollten daher keine an der Entwicklung des zu testenden Systems unmittelbar beteiligten Personen eingesetzt werden. Es wird sogar verschiedentlich empfohlen, daß der Testleiter sich völlig passiv verhält und dies dem Benutzer so auch vorher erklärt.

» Betrachten wir die Technik also als eine Option, die es uns gestattet, unsere Lebens- und Arbeitsräume menschengerecht und lebenswert zu gestalten. «

Während der Durchführung eines Benutzertests wird man immer wieder überraschende und sehr informative Benutzungsweisen (›kritische Vorfälle‹, ›Stolpersteine‹) beobachten können. Eine Videoaufzeichnung lohnt sich also, um diese dann später mit den EntwicklerInnen detailliert diskutieren zu können. Wichtig ist dabei, daß die zu beobachtenden Schwierigkeiten niemals ***dem Benutzer***, sondern ausschließlich ***der Benutzbarkeit*** des zu testenden Systems angelastet werden! Hatten mehrere Benutzer dieselben Schwierigkeiten, kann es an ihnen nicht liegen!

Benutzertests können also eine wichtige Methode sein, um dafür zu sorgen, daß Software zu einem echten Hilfsmittel bei der Aufgabenerfüllung wird. Am wichtigsten ist es jedoch, daß wir anfangen zu lernen, Technik, Organisation und den Einsatz menschlicher Qualifikation gemeinsam zu planen. Betrachten wir die Technik also als eine Option, die es uns gestattet, unsere Lebens- und Arbeitsräume menschengerecht und lebenswert zu gestalten.

Matthias Rauterberg
Institut für Arbeitspsychologie
Eidgenössische Technische Hochschule
Nelkenstraße 11, CH-8092 Zürich

Zur Vertiefung:

M. Rauterberg, P. Spinas, O. Strohm, D. Waeber, E. Ulich; Benutzerorientierte Software-Entwicklung – Konzepte, Methoden und Vorgehen zur Benutzerbeteiligung; Verein der Fachverlage, Zürich 1994