Dr. Ernest Wallmüller, Zürich, München, Wien
Qualität & Informatik
info@itq.ch

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Lektorat: Margarete Metzger
Herstellung: Irene Weilhart
Copy editing: Manfred Sommer, München
Umschlagdesign: Marc Müller-Bremer, www.rebranding.de, München
Umschlagrealisation: Stephan Rönigk
Datenbelichtung, Druck und Bindung: Kösel, Krugzell
Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702
Printed in Germany

print-ISBN: 978-3-446-40405-2
e-book-ISBN: 978-3-446-43019-8

Fortschritte im Usability Engineering

Matthias Rauterberg

Mit der fortschreitenden Automatisierung von Arbeitsprozessen werden verschiedene Ziele verfolgt. Beispielhaft seien hier genannt die Steigerung von Effizienz und Produktivität, die Verbesserung von Dienstleistungen, die Erhöhung von Wirtschaftlichkeit und Konkurrenzfähigkeit, die Verkürzung der Durchlaufzeiten von Werkstücken und Dokumenten, die Verbesserung von Arbeitsbedingungen, die Erhöhung der Auskunftsbereitschaft durch Integration von Informationen und verbesserten Zugriffsmöglichkeiten usw. Diese Ziele werden allerdings von den unterschiedlichen Interessengruppen im Betrieb – z. B. Management, Systementwickler, Organisatoren und Benutzer – vielfach unterschiedlich gewichtet, was zu Interessenkonflikten führen kann. Immer häufiger stellt sich deshalb bei Vorhaben zur Computerunterstützung von Arbeitstätigkeiten folgende Grundfrage: Wie können Anforderungen von Fachabteilungen bzw. Bedürfnisse von Benutzern nach Computerunterstützung von Arbeitsabläufen so in Software umgesetzt werden, dass diese den Anforderungen auch wirklich entspricht und dadurch eine echte Hilfe bei der Bewältigung der anfallenden Aufgaben darstellt? In den letzten Jahren ließen sich in vielen Software-Entwicklungsprojekten große Probleme feststellen, die manchmal zum Scheitern eines Projektes oder gar zum Konkurs einer Firma führten. Die Ursachen können vielfältiger Natur sein, denn für die Akzeptanz eines neuen Systems durch die Benutzer spielt die optimale Gestaltung folgender, unterschiedlicher Aspekte eine Rolle:

- Arbeitsteilung und -organisation,
- Mensch-Computer-Funktionsteilung,
- Benutzungsoberfläche (Usability),
- Ergonomie von Hardware sowie Arbeitsplatz und -umgebung,
- Entwicklungs- und Einführungsprozess mit Benutzerbeteiligung.

Von besonderer Bedeutung ist die Gestaltung der Arbeitsteilung und -organisation; sie hat vor bzw. zumindest gleichzeitig mit dem Einkauf oder der Entwicklung technischer Systeme zu erfolgen!

Bei den oben erwähnten Problemen bei der Einführung neuer Technologien handelte es sich aber meist weniger um technische Schwierigkeiten. Vielmehr ist es die fachliche Komplexität und Dynamik heutiger Anwendungen, die Software-Entwickler vor Probleme stellt. Ein ICT-Spezialist ist heutzutage häufig nicht mehr in der Lage, die fachliche Seite einer Anwendung hinreichend zu durchschauen und zu bewältigen. Deshalb ist eine enge Zusammenarbeit von Benutzern als Experten für das zu unterstützende Fachgebiet und Software-Entwicklern als Experten für Informatikwerkzeuge unabdingbar. Im Rahmen partizipativer Software-Entwicklung sind verschiedene Komponenten und Aspekte zu unterscheiden: Rahmenbedingungen, Projektmanagement, Aufbau- und Ablauforganisation, Information, Ausbildung, Partizipation sowie verschiedene Methoden zur Unterstützung von Benutzerbeteiligung.

Um Missverständnissen vorzubeugen, seien hier einige grundlegende Begriffe kurz erläutert:

- Unter Partizipation bzw. Beteiligung oder Einbezug von Benutzern ist deren Mitwirkung an der Planung und Entwicklung eines Systems und die Einflussnahme auf Entscheidungen zu verstehen.
- Als Benutzer oder Endbenutzer werden diejenigen Personen (Sachbearbeiter, Sekretärin, Manager usw.) bezeichnet, die das System nach abgeschlossener Entwicklung im Rahmen ihrer täglichen Arbeit benutzen. Im Gegensatz dazu ist mit Käufer der Betreiber oder Anwender gemeint (Organisation, Abteilung oder deren Repräsentanten).
- Bezeichnungen wie Entwickler, Designer, Programmierer werden in umfassendem Sinne synonym verwendet für Personen, die ICT-seitig an der Konzeption und Realisierung eines Systems beteiligt sind.

Abschließend sei noch auf folgende wichtige Unterscheidung hingewiesen: Wir haben es bei Software-Entwicklungsprojekten sowohl mit der Aufbau- und Ablauforganisation eines Unternehmens bzw. der betroffenen Fachabteilungen zu tun als auch mit der Aufbau- und Ablauforganisation des Software-Entwicklungsprojektes selbst. Diese Organisationsformen können unabhängig voneinander sein, aber auch weitgehend identisch. Wir haben diesem Umstand Rechnung getragen, indem wir die arbeitspsychologischen Kriterien für humane Arbeit, welche primär auf die Aufbau- und Ablauforganisation eines Unternehmens ausgerichtet sind, auch auf die Aufbau- und Ablauforganisation der Software-Entwicklung selbst übertragen und entsprechend angepasst haben.

Entgegen häufig vorgebrachten Argumenten – wie z. B.: die Benutzer wollen oder können gar nicht an einer Entwicklung beteiligt werden – ist die Bereitschaft zur Zusammenarbeit von Entwicklern und Benutzern nach den Ergebnissen meiner und anderer Untersuchungen viel größer, als gemeinhin angenommen. Ein Projektleiter formulierte es wie folgt: „Ohne die aktive Mitarbeit der Benutzer könnten wir heute keine Anwendung mehr realisieren. Der *erforderliche Zeitaufwand* ist im Verhältnis zu den *erzielten Vorteilen gering*." Es lassen sich vier Arten von Software-Entwicklungsprojekten unterscheiden: 1. spezifische Anwendungen für firmeninterne Fachabteilungen (Typ-A); 2. Spezifische Anwendungen für externe Kunden (Typ-B); 3. Standardbranchenlösungen für externe Kunden (Typ-C); 4. Standardsoftware für einen anonymen Kundenkreis (Typ-D). Während bei Typ-A bis C die potenziellen Benutzer weitgehend bekannt sind, müssen bei Typ-D neue Wege der Benutzerbeteiligung beschritten werden. Als besonderes Problem im Zusammenhang mit der Beteiligung von Benutzern bei Typ-D-Entwicklungen muss die große Heterogenität des Benutzerkreises angesehen werden. Um z. B. Usability-Tests durchführen zu können, sollte das zu testende System bestimmte Voraussetzungen erfüllen. Dies führt dazu, dass Usability-Tests vorwiegend im Entwicklungs- oder Lebenszyklus des Standardsoftware-Produktes eingesetzt werden. Um ein möglichst repräsentatives Abbild der verschiedenen Benutzungskontexte von Standardsoftware-Produkten zu erhalten, lassen sich Befragungen mittels Fragebögen unter dem registrierten Benutzerkreis sowie Contextual Inquiries durchführen. In einer verkürzten Form können auch wichtige Informationen von TestpartnerInnen offline über eine entsprechend erweiterte Version der Registrierkarten oder online über automatische Login-Techniken fortlaufend erhalten werden. Um End-Benutzer möglichst direkt und unmittelbar einzubeziehen, können Benutzer-Entwickler-Workshops durchgeführt werden. Bei reinen Neu-Entwicklungen lassen sich im Rahmen von Marktanalysen auch stichprobenartig Befragungen (mittels Fragebogen oder Interview) unter dem potenziellen Benutzerkreis durchführen. Als neuartige Methode hat sich die Beschrei-

bung von prototypischen End-Benutzern in Form von *Personas* mit typischen Benutzungsszenarien bewährt. Diese Szenarien können dann mit potenziellen, repräsentativ ausgewählten End-Benutzern besprochen, verbessert und somit validiert werden. Eine weitgehend vollständige Übersicht über einschlägige Usability-Methoden ist via Internet verfügbar: http://www.usability.gov/methods/index.html.

Besonders hervorzuheben ist, dass die Projektleitung sich mit den Zielen und Inhalten des Projektes identifizieren muss, damit sie das Projekt wirkungsvoll und glaubwürdig als Promotor vertreten kann. Unter Empathie ist Einfühlungsvermögen und eine offene Wahrnehmung zu verstehen für das, was im Projekt geschehen kann, real geschieht oder auch nicht geschieht. Mögliche Gefahren sollten der Projektleitung bekannt sein und entsprechende Warnsignale von ihr frühzeitig, richtig und vollständig erkannt und gedeutet werden. Aus verschiedenen Erfahrungsberichten von Software-Projekten sind vor allem folgende Gefahren und Warnsignale bekannt:

(1) Man überschätzt sich mit dem Projekt bzw. nimmt sich zu viel vor.

(2) Die Projektziele sind unklar oder schließen sich gegenseitig aus.

(5) Dem Projekt fehlt ein (Macht-)Promotor auf der Anwendungsseite.

(6) Die Komplexität der fachtechnischen Aufgabenstellung wird unterschätzt.

(7) Das Projekt ist „angeordnet" und vom Anwendungsbereich nicht gewollt.

(8) Das Projekt wird kosten- und terminbezogen viel zu knapp budgetiert.

(9) Die organisatorischen Gegebenheiten im Anwendungsbereich werden vor der Systemkonzeption nicht hinterfragt.

(10) Es fehlen angemessene Methoden und Werkzeuge (z. B. software-technische Tools, Prototyping-Techniken, Zugang zu ISO-Normen und anderer relevanter Literatur, ein interaktives und benutzungszentriertes Vorgehensmodell, etc.).

(11) Der Zeitplan „gerät schon sehr früh ins Wanken".

(12) Die Qualifikationen der Mitglieder im Team sind unzureichend.

(13) Die Ergebnisse werden unzureichend dokumentiert.

(14) Der Software-Entwurf lässt keine nachträglichen Anforderungsänderungen zu.

(15) In der Projektorganisation fehlen die Benutzervertreter.

(16) Die Qualitätssicherung wird vernachlässigt.

(17) Der Projektgruppe fehlt es an Motivation.

Die meisten dieser Gefahren lassen sich durch eine entsprechende Projektorganisation vermeiden. Wichtig ist, dass alle Beteiligten von einem benutzungszentrierten Vorgehen überzeugt sind. Um nun den End-BenutzerInnen die Möglichkeit zu geben, ihre wertvollen Fachkenntnisse sinnvoll einzubringen, hat sich das Prototyping besonders vorteilhaft herausgestellt.

Während in der Produktgüterindustrie ein Prototyp in der Regel ein voll funktionstüchtiges und fehlerfreies Produkt darstellt, ist ein Prototyp im Bereich Software ein irgendwie unfertiges Produkt: sei es, dass die eigentliche Funktionalität oder die eigentliche Oberfläche fehlt, sei es, dass das Antwortzeitverhalten unzureichend oder die interne Programmstruk-

tur nicht nach gängigen softwaretechnischen Kriterien ausgestaltet ist. Auf alle Fälle sollte ein Software-Prototyp operational sein; ob diese Operationalität direkt gegeben sein muss oder auch simuliert werden kann, hängt vom Einzelfall ab. Software-Prototypen sollten im Rahmen einer Prototyping-Strategie schnell und kostengünstig realisiert werden können. Prototyping ist im Kontext partizipativer Software-Entwicklung eine Spezifikationsmethode, bei der die softwareergonomischen Anforderungen zur Benutzungsoberfläche präzise festgelegt werden können. Darüber hinaus kann ein Prototyp zur Erprobung und Evaluation von Lösungsvorschlägen, zur Beurteilung der Qualität des Entwurfs einer Systemarchitektur und zur Prüfung verschiedener Realisierungsmöglichkeiten von Systemkomponenten eingesetzt werden.

Während ein *horizontaler* Prototyp eine weitgehend funktionslose Fassade („mock up") darstellt, umfasst ein *vertikaler* Prototyp nur das rein funktionale Modell ohne die angestrebte Benutzungsoberfläche. Horizontale Prototypen eignen sich hervorragend für Benutzer-Entwickler- Diskussionen. Vertikale Prototypen dagegen eignen sich eher zum Austesten von Anwendungsfunktionen im Rahmen von Machbarkeitsstudien. Wenn ein zunächst unvollständiger Prototyp zum vollständigen Zielsystem ausgebaut wird, spricht man von einer evolutionären Prototyping-Strategie. Mit immer mächtiger werdenden Entwicklungsumgebungen kann Prototyping fließend in die eigentliche Systementwicklung übergehen. Prototyping als benutzernahe Methode für den Software-Entwicklungsprozess hat inzwischen eine breite Verwendung gefunden und wird von vielen Unternehmen schon regelmäßig eingesetzt.

Mit dem systematischen Einbezug von Benutzern in den Entwicklungsprozess werden primär folgende Ziele verfolgt:

- *Innovation*
  Es werden innovativere Lösungen entwickelt, in welche neben dem technischen Wissen der Entwickler auch das Fachwissen der Benutzer einfließt.
- *Identifikation*
  Die Entwickler wie auch die Benutzer können sich mit der Entwicklungsarbeit sowie mit der Lösung besser identifizieren.

Zur Erreichung dieser Ziele müssen folgende Grundbedingungen erfüllt sein:

- *Integration*
  Aufgabenspezifische, benutzerspezifische und technische Anforderungen müssen *integrativ* analysiert und in der Umsetzung berücksichtigt werden.
- *Interaktion*
  Entwickler und Benutzer müssen die Möglichkeit haben, direkt miteinander zu interagieren, damit sich ein gegenseitiges Verständnis entwickeln kann.
- *Iteration*
  Der Entwicklungsprozess muss so angelegt sein, dass innerhalb einzelner Phasen wie auch über Phasen hinweg iteriert werden kann. Nur so können Veränderungen in den Anforderungen aufgefangen, getroffene Entscheidungen überprüft, Gestaltungsvorschläge, Konzepte evaluiert und gegebenenfalls frühzeitig korrigiert werden.

Inzwischen haben auch namhafte Hersteller sowie Anwender mit eigener Entwicklungsabteilung die Vorteile des Einbezugs von Endbenutzern erkannt. In ähnlicher Weise wird der Benutzereinbezug auch bei verschiedenen Unternehmen in Projekthandbüchern ver-

bindlich festgelegt. Diese Beispiele zeigen, dass Benutzerbeteiligung kein theoretisches Konzept mehr, sondern inzwischen praktische Realität ist!

Prof. Dr. Matthias Rauterberg ist Diplom-Informatiker und Diplom-Psychologe mit Forschungsschwerpunkten in benutzerzentriertem Design, Benutzermodellierung und Entertainment Computing. Schon vor über 20 Jahren hat er empirische Benutzertests in unterschiedlichsten Software-Entwicklungsprojekten erfolgreich entwickelt und eingesetzt. Seit 1998 ist er Professor für „Human Communication Technology" und Forschungsgruppenleiter am Fachbereich Industrial Design der Eindhoven University of Technology, Holland. ■