
# ErgoNet - ein netzbasiertes, multimediales System zur Unterstützung von Ad-hoc Usability-Tests

Diplomarbeit

**Carsten Leßmann**

**Manfred Wolff**


Universität Bremen

Fachbereich 3 Mathematik/Informatik

Studiengang Informatik

Bremen, im Januar 1999

# Inhaltsverzeichnis

# 1 Einleitung

**Bedeutung ergonomischer Software**

In dieser Arbeit befassen wir uns mit der Evaluation von Software im Bereich der Software-Ergonomie. Seitdem nicht mehr nur EDV-Fachleute mit Software umgehen müssen, spielt die ergonomische Gestaltung von Benutzungsoberflächen eine immer größere Rolle. Mit dem Inkrafttreten der Bildschirmarbeitsverordnung gibt es ein Recht der BenutzerInnen[1] auf ergonomisch gut gestaltete Benutzungsschnittstellen zur Software. Im besonderen mit der Normung der DIN EN ISO 9241 verfügen wir über ein Gerüst, um softwareergonomisch anspruchsvolle Software zu erstellen und über klar definierte Kriterien, Software zu evaluieren.

**Evaluation in frühen Phasen der Softwareentwicklung**

Software-ergonomische Evaluationen werden im wesentlichen am Ende des Software-Entwicklungszyklus durchgeführt. Die Behebung der Mängel zu diesem Zeitpunkt ist jedoch deutlich zeit- und ressourcenaufwendiger, als wenn sie bereits in frühen Phasen der Softwareentwicklung durchgeführt würde. Benötigt werden Mittel und Methoden, mit deren Hilfe Fehler in der Benutzungsschnittstelle in jedem Stadium der Softwareentwicklung gefunden und behoben werden können.

**Ziel der Arbeit**

Ziel dieser Arbeit ist es, ein System zu entwickeln, mit dessen Hilfe softwareergonomische Evaluationen effektiv und effizient während des gesamten Software-Entwicklungszyklus durchgeführt werden können. Wir entwickeln einen vertikalen Prototyp, mit dem Usability-Tests in räumlicher Distanz im Netz durchgeführt werden können. Usability-Tests die bilden eine Säule bei Evaluationsmaßnahmen, expertInnengestützte Tests (*Reviews*) die andere. Die Diplomarbeit setzt sich aus drei Teilen zusammen, die gleichzeitig Teilschritte unserer Systemerstellung beschreiben.

**Darstellung von Evaluationsmethoden**

In dem ersten Schritt beschäftigen wir uns mit Evaluationsmethoden von Benutzungsoberflächen (Kapitel 2). Besonders im US-amerikanischen Umfeld sind in den letzten Jahren eine Reihe von Methoden entwickelt und in den jährlichen Tagungen des ACM (*Association for Computing Machinery*) diskutiert und ausgewertet worden. Wir prüfen diese Methoden auf ihre Effizienz, Effektivität und Eignung für unser System. Wir untersuchen, welche Voraussetzungen gegeben sein müssen, um Usability-Tests effektiv und ohne großen Aufwand über das Netz durchzuführen (Kapitel 3).

**Ermittlung von Anforderungen, Spezifikation**

Der zweite Schritt ist die Anforderungsermittlung (Kapitel 4), die Umsetzung (Kapitel 5) und die Implementierung (Kapitel 6) eines Softwaresystems zur Unterstützung von Usability-Tests im Netz. Sowohl die Anforderungsermittlung und Spezifikationserstellung als auch die Implementierung erfolgt iterativ mit Hilfe von prototypischen Verfahren. In drei Iterationen entwickeln wir ein Basissystem, mit dem Tests durchgeführt und die Ergebnisse dokumentiert werden können. Darüber hinaus diskutieren wir mögliche Weiterentwicklung des Systems.

[1] Wenn wir von Menschen beiderlei Geschlechts schreiben, benutzen wir eine Schreibweise, wie sie z. B. die TAZ verwendet. Wenn wir BenutzerInnen schreiben, meinen wir Benutzer und Benutzerinnen. Wo immer es ging, haben wir geschlechtsneutrale Begriffe gewählt (z. B. die Studierenden für die Studentinnen und Studenten).

**Erprobung des Softwaresystems**

Die Erprobung und Präsentation unserer Software bildet den dritten Schritt (Kapitel 7). In vier verschiedenen Erprobungen bzw. Präsentationen sollen Praxistests die Qualität der Software prüfen. Nach jedem Schritt werden Schlußfolgerungen für die Weiterentwicklung der Software gezogen. Zwei der Tests fanden während der CeBIT98 und dem Tag der offenen Tür des TZI[2] statt. Die beiden anderen Erprobungen erfolgten in Zusammenarbeit mit der Bremer Schule Bürgerweide (BSB) und dem Kohne Ingenieurbüro GmbH mit realen Softwareentwicklungen.

**Resümee und Anhang**

Im Schlußteil (Kapitel 8) ziehen wir ein Resümee unserer Arbeit. Im Anhang befinden sich Dokumente, die als Vorarbeiten zu dieser Arbeit dienten. Auf CD befindet sich Literatur, die wir aus dem WorldWideWeb bezogen haben sowie die Quelltexte und eine Installationsversion des Systems ErgoNet.

[2] Technologiezentrum Informatik in der Universität Bremen.

# 2 Leistungsfähigkeit und Grenzen softwareergonomischer Evaluationsmethoden

**Zwei Schnittstellen von Anwendungssoftware**

Jede Anwendungssoftware hat zwei Schnittstellen: eine zur Maschine (zum Rechner) und eine zum Menschen. In dem Maße, wie die Bedienung von Software die Kommandozeile verlassen hat, spielt die Schnittstelle zum Menschen eine immer größere Rolle. Die Software muß an die Bedürfnisse und Anforderungen der Menschen so genau anpaßt werden wie an die sich ständig ändernde Hardware (vgl. HERING 1992).

**Ziel von Kapitel 2**

Wir befassen uns in diesem Kapitel mit der Schnittstelle Software - Mensch. Wir beleuchten qualitative Anforderungen an die Benutzungsoberfläche und befassen uns mit deren Evaluierung. Wir widmen uns einem Widerspruch: In Zeiten, in denen die Benutzungsoberfläche eine immer größere Bedeutung erhält, steigt nicht im gleichen Maße der Aufwand für die Evaluation von Software - auf der Strecke bleibt die Qualität.

## 2.1 Die Bedeutung der Software-Ergonomie

**Verschiedene Ansätze in der Software-Ergonomie**

Ergonomie ist die „Wissenschaft von den Leistungsmöglichkeiten des arbeitenden Menschen und der Anpassung der Arbeitsbedingungen an den Menschen“ (MEYERS LEXIKON 1997). In der Ergonomie gibt es verschiedene Ansätze, die wie folgt zu charakterisieren sind (vgl. PAUL 1995):

**Der wirtschaftlich-technische Ansatz**

1. **Der wirtschaftlich-technische Ansatz**
   Bei diesem Ansatz wird nach einer optimalen Übereinstimmung zwischen Arbeitsproduktivität und zumutbarer Arbeitsbeanspruchung gesucht. Der Mensch ist wie die Technik eine Variable, eine veränderbare Größe.

**Der anthroprozentrische Ansatz**

2. **Der anthroprozentrische Ansatz**
   Dieser Ansatz geht davon aus, daß die Technik dem Menschen angepaßt werden muß. AnhängerInnen dieses Ansatzes gestalten aber nicht nur die Technik, sondern darüber hinaus auch das Umfeld, in dem Technik eingesetzt wird, z. B. die Ablauforganisation.

**Sichtweisen in der Software-Ergonomie**

Eine ähnliche Unterscheidung wie bei der Ergonomie findet sich auch in der Software-Ergonomie. Die eine Sichtweise konzentriert sich dabei nur auf die Benutzungsoberfläche (*user interface*[3]), während sich die zweite Sichtweise auf die Anpassung des gesamten DV-Systems konzentriert. Dabei geht es vor allem um Faktoren, die die Arbeitssituation von ComputerbenutzerInnen beeinflussen: Mensch, Aufgabe, Technik und organisatorischer Rahmen (vgl. MAAß 1993 S. 191). Folgt man diesem Ansatz in der Software-Ergonomie, so hat dies auch Konsequenzen für die Evaluation von Software. Wir werden darauf in Kapitel 2.3 näher eingehen.

[3] Da wir im wesentlichen mit englischsprachiger Literatur gearbeitet haben, setzen wir bei Definitionen das englische Wort in Klammern. Wir benutzen danach das englische und deutsche Wort synonym. Wo wir können, übersetzen wir sinngemäß. Englische Begriffe sind *kursiv* gesetzt.

**Drei Schwerpunkte der Software-Ergonomie**

Software-Ergonomie hat sich als interdisziplinäre Wissenschaft etabliert. Im wesentlichen sind drei Schwerpunkte auszumachen: Der technische, der kognitiv-psychologische und der arbeitspsychologische Schwerpunkt (vgl. MAAß 1993, MAAß 1995):

**Der technische Schwerpunkt**

1. **Der technische Schwerpunkt**
   InformatikerInnen und SystementwicklerInnen suchten lange Zeit Lösungen für eine benutzungsfreundliche Systemgestaltung im technischen Bereich. Auswirkungen der Forschungen mit technischem Schwerpunkt sind beispielsweise Funktionstasten für die Reduzierung des Schreibaufwands, neue Eingabegeräte wie Maus und Lichtgriffel, neue Dialogarten wie das Prinzip der direkten Manipulation. Der technische Schwerpunkt ist derjenige, der sowohl in der Wissenschaft als auch in der Praxis am meisten beachtet wurde.

**Der kognitiv-psychologische Schwerpunkt**

2. **Der kognitiv-psychologische Schwerpunkt**
   Parallel zum technischen Schwerpunkt untersuchten Psychologen die kognitiven Aspekte der Mensch-Rechner-Interaktion. Auswirkungen dieser Forschungsrichtung waren z. B. Regeln für den Aufbau von Bildschirmmasken, Tiefe von Menübäumen und Piktogramme. Insbesondere wird die Explorationsfreudigkeit von Softwaresystemen angeregt (vgl. PAUL 1995). Dieser Schwerpunkt wurde vor allem in den USA entwickelt (u. a. von Norman, Polson und Lewis).

**Der arbeitspsychologische Schwerpunkt**

3. **Der arbeitspsychologische Schwerpunkt**
   Bei dem arbeitspsychologischen Schwerpunkt spielen Kriterien wie Zumutbarkeit, Schädigungsfreiheit, Beeinträchtigungslosigkeit und Persönlichkeitsförderlichkeit eine große Rolle. Diese Sichtweise ist eher in Europa beheimatet. Begründer dieser Theorien sind die Arbeitspsychologen Hacker, Ulich und Volpert. Softwaregestalter sind in diesem Sinne vor allem Arbeitsorganisatoren (vgl. HACKER 1994). Diese Theorie bietet auch den Rahmen für die speziellen softwareergonomischen Forderungen in der DIN 66 234 Teil 8.

**Regionale Schwerpunkte**

Wie bei der Erläuterung der verschiedenen Forschungsschwerpunkte bereits angedeutet, gibt es im Bereich der Software-Ergonomie verschiedene regionale Schwerpunkte, die sich auch im Verlauf der Arbeit weiter zeigen werden. Die kognitiv-psychologische Forschung findet hauptsächlich in Großbritannien und in den USA statt. Der arbeitspsychologische Ansatz kommt eher aus dem deutschsprachigen Raum. Partizipative Ansätze kommen aus Skandinavien und aus Deutschland. Die Idee des Usability-Tests kommt aus den USA, die in Forschung und Entwicklung praktischer Ansätze traditionell mehrere Jahre im Vorsprung sind (vgl. MAAß 1993).

**EU-Richtlinie**

Die praktische Bedeutung der Software-Ergonomie wird unter anderem in der EU-Richtlinie unterstrichen, die seit 1993 diejenigen, die computergestützte Arbeitssysteme einführen, verpflichtet, das gesicherte Wissen der Software-Ergonomie zu berücksichtigen (vgl. EU-RICHTLINIE 1990). Diese Richtlinie ist durch die „Verordnung über Sicherheit und Gesundheitsschutz bei der Arbeit an Bildschirmgeräten“ seit 20.12.1996 Gesetz. Detailliert heißt es dort im Kapitel „Zusammenwirken Mensch – Arbeitsmittel“:

„Bei Entwicklung, Auswahl, Erwerb und Änderung von Software sowie bei der Gestaltung der Tätigkeit an Bildschirmgeräten hat der Arbeitgeber den folgenden Grundsätzen, insbesondere im Hinblick auf die Benutzerfreundlichkeit, Rechnung zu tragen:

- Die Software muß an die auszuführende Tätigkeit angepaßt sein.
- Die Systeme müssen den Benutzern Angaben über die jeweiligen Dialogabläufe unmittelbar oder auf Verlangen machen.
- Die Systeme müssen dem Benutzer die Beeinflussung der jeweiligen Dialogabläufe ermöglichen sowie eventuelle Fehler bei der Handhabung beschreiben und deren Beseitigung mit begrenztem Arbeitsaufwand erlauben.
- Die Software muß entsprechend den Kenntnissen und Erfahrungen der Benutzer im Hinblick auf die auszuführende Aufgabe angepaßt werden können."

## 2.2 Qualitätsmerkmale in der Software-Ergonomie

**Bedarf an Qualitätsdefinitionen**

Das Ziel der Software-Ergonomie, „die Entwicklung und Gestaltung gut benutzbarer Computersysteme als intellektuelle Werkzeuge und die Verbesserung von Benutzungsschnittstellen oder Benutzungsoberflächen" (WANDMACHER 1993 S. 1) beinhaltet, daß die Qualität von benutzbaren Systemen definiert werden muß. Die Gestaltung gut benutzbarer Systeme darf nicht individuell von den jeweiligen EntwicklerInnen oder DesignerInnen betrachtet und bewertet werden.

**Erfahrungen anderer Wissenschaften**

In anderen gewachsenen technischen Bereichen (z. B. Architektur) wurde die Erfahrung gewonnen, daß die systematische Kodifikation erfolgreicher Praktiken eine Wiederholung bekannter Fehler verhindert. Die Inhalte dieser Bereiche, also die„Methoden (Zeichnungen und Entwürfe), Standards (für Fundamente, Rahmen, Komponenten), Werkzeuge und eine Vielfalt spezieller Arbeiten"[4] (GRADY 1993 S. 62), lassen sich auf das Gebiet der Software-Erstellung übertragen. Eine Standardisierung kann auch hier die Qualität der Produkte positiv beeinflussen, dies muß aber nicht zwingend die Folge sein (siehe Kapitel 2.2.5).

Weitere Gründe für die Notwendigkeit einer Standardisierung lassen sich in den folgenden vier Kategorien zusammenfassen (vgl. HOLDAWAY & BEVAN):

1. Konsistenzbedarf (*need for consistency*)
2. Gesteigerter Usability-Bedarf (*need for enhanced usability*)
3. BenutzerInnenbedürfnisse (*need for assurance of the user s comfort and well being*)
4. Produktbeschaffung und -evaluation (*procurement and product evaluation*)

**Technische vs. ergonomische Standards**

Im Gegensatz zu anderen technischen Bereichen, deren Standards eine präzise technische Spezifikation sind (vgl. BEVAN 1995 S. 1, BEIMEL, SCHINDLER &

[4] Eigenübersetzung.

WANDKE 1993 S. 134), würden restriktive Normen die „gestalterische Freiheit“ [5] (CAKIR & DZIDA 1997 S. 407) der EntwicklerInnen und DesignerInnen verhindern. Darüber hinaus veralten ergonomische Normen dieser Art mit jedem neuen technologischen Entwicklungsschritt und können sich nur auf restringierte BenutzerInnentypen und -aufgaben beziehen. Aus diesem Grunde befassen sich die meisten Arbeiten im Bereich der Software-Ergonomie nicht mit präzisen Spezifikationen, sie „konzentrieren sich auf die grundsätzlichen Prinzipien, die angewendet werden müssen, um eine benutzerInnen- und aufgabenangemessene Oberfläche zu erstellen.“ [6] (BEVAN 1995 S. 1).

### 2.2.1 Was ist Qualität?

**Verschiedene Klassifikationen für Qualität**

Die Bedeutung des Begriffes Qualität ist im alltäglichen Gebrauch selbsterklärend. Im Bereich der Software-Entwicklung gibt es dagegen verschiedene Sichtweisen für Qualität und unterschiedliche Verfahren, sie herzustellen. Die folgende Aufzählung gibt einen Überblick über eine mögliche Klassifizierung für Qualität (vgl. BEVAN 1994 S. 2ff, BEVAN S. 1):

1. **Produktbezogene Qualität (*product quality*)**
   Die innewohnenden Charakteristiken des Produkt, die mit der An- oder Abwesenheit meßbarer Produktattribute bestimmt werden.
2. **Herstellungsbezogene Qualität (*manufacturing quality*)**
   Das Produkt, das mit bestimmten Anforderungen übereinstimmt.
3. **BenutzerInnenbezogenene Qualität (*user perceived quality*)**
   Die Kombination von Produktattributen, mit der die höchste Zufriedenheit der spezifizierten BenutzerInnen erreichtwird.
4. **Ökonomische Qualität (*economic quality*)**
   Das Produkt, das Leistung zu einem akzeptablen Preis oder Konformität von Anforderungen zu akzeptablen Kosten vorsieht.

**Externe Qualität (external quality)**

Die meisten dieser Ansätze beziehen sich hauptsächlich auf das Produkt und seine Eigenschaften, ohne den eigentlichen Einsatzzweck zu berücksichtigen. Dies führt zu einer ausgedehnteren Definition von Qualität, die auch den Zweck des Produktes berücksichtigt. „Produkte können nur Qualität in Relation zu ihrem beabsichtigten Zweck besitzen“ [7] (BEVAN S. 1), so daß der Fokus für Qualität vom Produkt an sich zu den BenutzerInnen des Produktes verschoben werden muß. Dies führt zum Begriff Externe Qualität, der „Qualität als den Umfang definiert, in dem ein Produkt implizite und explizite Bedürfnisse unter spezifizierten Konditionen erfüllt“[8] (BEVAN S. 1).

**Gebrauchsqualität (quality in use)**

Die BenutzerInnen und den Einsatzzweck des Produktes bei der Qualitätsbeurteilung eines Produktes hinzuziehen, führt zu einer noch offeneren Sichtweise und Beurteilung von Produkten, nämlich der Gebrauchsqualität. Unter Gebrauchsqualität versteht man die „Effektivität, Effizienz und Zufriedenheit,

[5] *freedom in design.*
[6] Eigenübersetzung.
[7] Eigenübersetzung.
[8] Eigenübersetzung.

mit der ein spezifischer Benutzer aufgeführte Ziele in einer definierten Umgebung erreichen kann."[9] (BEVAN S. 2).

**Gebrauchsqualität und Softwarequalität**

„Der Zweck des Designs von interaktiven Systemen ist es, die Bedürfnisse der BenutzerInnen zu erreichen. Die internen Software-Attribute entscheiden über die Qualität eines Software-Produktes während des Einsatzes in einem bestimmten Kontext. Softwarequalitäts-Attribute sind die Ursache, während die Gebrauchsqualität der Effekt ist. Gebrauchsqualität ist (oder sollte es zumindest sein) das Ziel, Softwareprodukt-Qualität ist das Mittel, um sie zu erreichen."[10] (BEVAN S. 2).

Diese Sichtweise eines Produktes und seiner Eigenschaften führt bei genauerer Betrachtung zum Begriff Usability.

### 2.2.2 Was ist Usability?

**Gebrauchsqualität führt zu Usability**

Gebrauchsqualität läßt sich von allen Qualitätscharakteristiken eines Produktes beeinflussen. Im Gegensatz dazu bezieht sich Usability per Definition nur auf die folgenden Bereiche und ist damit eingeschränkter als Gebrauchsqualität (vgl. BEVAN S. 4):

1. Verständlichkeit (*understandibility*)
2. Erlernbarkeit (*learnability*)
3. Bedienbarkeit (*operability*)

**Verschiedene Sichtweisen von Usability**

Usability läßt sich, ähnlich zu Qualität, in folgende Klassifikationen einteilen (vgl. BEVAN, KIRAKOWSKI & MAISSEL 1991 S. 1):

1. **Produktbezogene Usability (*product-oriented view*)**
   Usability kann in bezug auf die ergonomischen Attribute gemessen werden.
2. **BenutzerInnen-bezogene Usability (*user-oriented view*)**
   Usability kann in bezug auf die mentale Belastung und Einstellung der BenutzerInnen gemessen werden.
3. **Leistungsbezogene Usability (*user performance view*)**
   Usability kann gemessen werden, indem die BenutzerInnen beobachtet werden, während sie mit dem Produkt interagieren.

Diese Einteilung wird durch folgende Definition vervollständigt, die den Gebrauch, die BenutzerIn und den Kontext berücksichtigt (vgl. BEVAN, KIRAKOWSKI & MAISSEL 1991 S. 2, KARAT J. 1997 S. 691):

**Umfassende Definition von Usability**

„Usability ist die Effektivität, die Effizienz und Zufriedenheit, mit der eine spezifische BenutzerIn aufgeführte Ziele in einer definierten Umgebung erreichen kann."[11]

**Usability ⇔ Gebrauchsqualität**

Usability hat damit die gleiche Definition wie Gebrauchsqualität und kann im weiteren äquivalent betrachtet werden.

---

[9] Eigenübersetzung.
[10] Eigenübersetzung.
[11] Eigenübersetzung.

Diese sehr allgemein gehaltenen Merkmale von Usability werden im weiteren genauer ausgeführt, da Usability den zentralen Gegenstand dieser Diplomarbeit darstellt.

### 2.2.3 Zusammenhänge und Faktoren von Usability

**Zusammenhänge der Usability**

Folgende Grafik verdeutlicht die komplexen Zusammenhänge zwischen den BenutzerInnen, den Aufgaben und der physikalischen und sozialen Umgebung, in denen Usability betrachtet und bewertet wird (vgl. BEVAN, KIRAKOWSKI & MAISSEL 1991 S. 3):

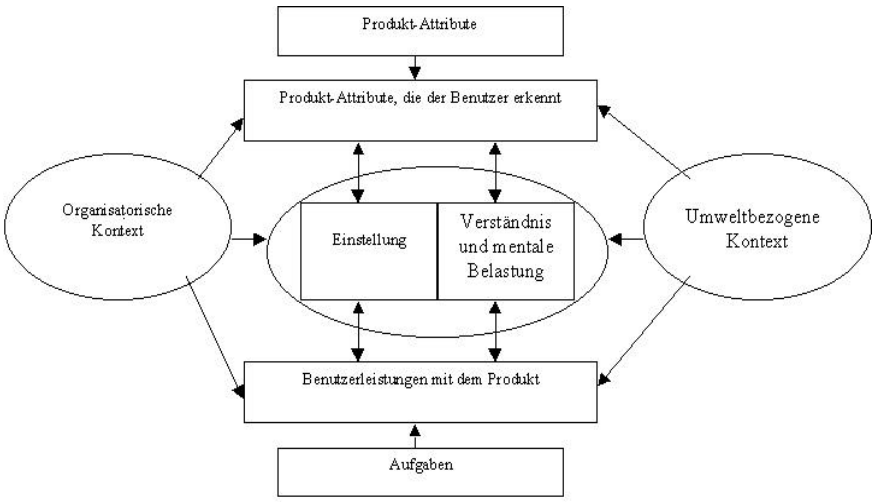



Abbildung 1: Usability-Determinaten

**Eigenschaften der Usability**

Die nachfolgenden fünf Eigenschaften gelten als die klassischen Attribute, die im Zusammenhang mit Usability betrachtet werden (vgl. NIELSEN 1993 S. 26ff). Sie stellen eine praktische Verfeinerung der Bereiche dar, die für Usability gelten:

1. **Erlernbarkeit (*learnability*)**
   Das System sollte leicht erlernbar sein, damit die BenutzerInnen schnell ihre Arbeit mit dem System erledigen können.

2. **Effizienz (*efficiency*)**
   Das System sollte leistungsfähig sein, damit die BenutzerInnen eine hohe Produktivität erreichen können, nachdem die Benutzung des Systems erlernt wurde.

3. **Einprägsamkeit** (**memorability**)
   Das System sollte einprägsam sein, damit gelegentliche BenutzerInnen das System auch nach längerer Nicht-Benutzung wieder einsetzen können, ohne alles neu erlernen zu müssen.

4. **Fehler (errors)**
   Das System sollte eine geringe Fehlerrate besitzen, d. h. ‚katastrophale Fehler' sollten nicht vorkommen und Fehlhandlungen der BenutzerInnen dürfen nicht provoziert bzw. noch so verstärkt werden, daß ein Weiterarbeiten unmöglich wird.

5. **Zufriedenheit (satisfaction)**
   Das System sollte angenehm zu benutzen sein, damit die BenutzerInnen subjektiv zufrieden sind.

Mit diesen Kriterien kann die Usability eines Produktes betrachtet und bewertet werden.

### 2.2.4 Standards

**Sachverhalte der ergonomischen Standards**

Um ergonomische Standards zu etablieren, die weder die gestalterische Freiheit der DesignerInnen und EntwicklerInnen beschneiden noch in der Abhängigkeit eines technologischen Entwicklungszyklus stehen, werden keine restriktiven Regeln aufgestellt, sondern folgende Bereiche standardisiert:

- **Das Vorgehen**
  z. B. das Vorgehen einer Organisation, um die Qualität eines Produktes zu sichern
- **Die Mindestanforderungen an Performance**
  z. B. Geschwindigkeit und Genauigkeit einer spezieller Tastatur
- **Die Methoden zur Messung bestimmter Charakteristiken**
  z. B. Belastung in einer Umgebung.

Im weiteren folgt eine Auflistung der Standards, die Usability als Gestaltungskontext hat (vgl. BEVAN 1995, BEVAN 1997, BEVAN & CURSON, HOLDAWAY & BEVAN, CAKIR & DZIDA 1997). Sie bilden zusammengenommen einen kompakten Komplex, der den Bereich der Usability umfassend beinhaltet:

Die verwendeten Begriffe werden definiert (ISO 8402); es wird ein benutzerInnen-orientierter Prozeß beschrieben (ISO 13407), der die Regelungen der DIN 66234 bzw. ISO 9241 berücksichtigen muß. Die Software-Qualitäts-Charakteristiken werden in der ISO 9126 erläutert, während die Evaluation in der ISO 14598 erörtert wird.

1. **ISO CD 13407 *User centred design process for interactive systems***
   **ISO CD 13407 User centred design process for interactive systems**
   Dieser Standard beinhaltet eine prozeß-orientierte Annäherung an Usability, mit der benutzbare Systeme als Resultat eines benutzerInnenzentrierten Designprozesses erreicht werden.

2. **ISO DIS 9241 *Ergonomic requirements for office work with visual display terminals***
   **ISO DIS 9241 Ergonomic requirements for office work with visual display terminals**
   - Part 1 *General introduction*
     Generelle Einführung in den Multistandard ISO 9241 und Beschreibung der Struktur dieses Standards, der aus den folgenden drei Gruppen besteht:
     Gruppe 1: Arbeit, Organisation und ihre Rollen in bezug auf Usability (Part 2, Part 11)
     Gruppe 2: Arbeitsplatz und Arbeitsumgebung (Part 5 - 6)
     Gruppe 3: Interaktives Equipment/Werkzeuge
     - Visuelle Anzeigen:
       Part 3, Part 7
     - Eingabe-Instrumente:

Part 4, Part 9
- Software-Oberflächen:
  Part 10, Part 12 - 17
- Farben/Visuelle Anzeigen/Software-Oberflächen:
  Part 8

- Part 2 *Guidance on task requirements*
  Bezieht ergonomische Prinzipien auf das Design von Benutzer-Innenaufgaben mit visuellen Anzeigeterminals.

- Part 3 *Visual display requirements*
  Formuliert Leistungsanforderungen und spezifiziert Konformitäts-Tests von visuellen Anzeigeterminals (speziell monochrome Anzeigen; siehe auch Part 8).

- Part 4 *Keyboard requirements*
  Befaßt sich mit ergonomischen Aspekten von alphanumerischen Tastatur-Designs.

- Part 5 *Workstation layout and postural requirement*
  Definiert ergonomische Anforderungen an visuelle Anzeigeterminals, damit BenutzerInnen eine komfortable und effiziente Körperhaltung annehmen können.

- Part 6 *Environmental requirements*
  Formuliert Grundsätze in bezug auf Umgebungseinflüsse (Licht, Lärm, etc.), die erfüllt werden müssen.

- Part 7 *Display requirements with reflections*
  Formuliert Leistungsanforderungen und spezifiziert Konformitäts-Tests von visuellen Anzeigeterminals in bezug auf Lichtreflektionen und -einstrahlungen.

- Part 8 *Requirements for displayed colours*
  Definiert den ergonomischen Einsatz von Farbe bei graphischen Anzeigeterminals, die hier aufgezeigten Prinzipien können auch auf Software übertragen werden.

- Part 9 *Requirements for non-keyboard input devices*
  Befaßt sich mit den ergonomischen Anforderungen an alle Eingabeinstrumente außer der Tastatur (siehe auch Part 4).

- Part 10 *Dialogue principles*
  Definiert generelle ergonomische Prinzipien, die sich mit dem Design des Dialoges zwischen Menschen und Informationssystemen beschäftigen (Grundlage bildet DIN 66234-Part 8).

- Part 11 *Guidance on Usability*
  Definiert den Begriff Usability und betont, daß Usability auf dem Gebrauchskontext basieren muß (siehe oben).

- Part 12 *Presentation of information*
  Enthält spezifische Empfehlungen für die Präsentation und Darstellung von Informationen auf visuellen Displays.

- Part 13 *User guidance*
  Enthält Empfehlungen für das Design und die Evaluation benutzerInnenbezogener Anleitungen für Software-Oberflächen.
- Part 14 *Menu dialogues*
  Enthält Empfehlungen für das ergonomische Design von Menüs in BenutzerInnen-Computer-Dialogen.
- Part 15 *Command language dialogues*
  Enthält Empfehlungen für das ergonomische Design von Kommando-Sprachen in BenutzerInnen-Computer-Dialogen.
- Part 16 *Direct manipulation dialogues*
  Enthält Empfehlungen für das ergonomische Design von direktmanipulativen Dialogen und Objekten.
- Part 17 *Form-filling dialogues*
  Enthält Empfehlungen für das ergonomische Design formularfüllender Dialoge.

3. **ISO 8402 *Quality Vocabulary*** — **ISO 8402 Quality Vocabulary**

4. **ISO DIS 14598 *Information Technology-Software product evaluation*** — **ISO DIS 14598 Information Technology-Software product evaluation**
   - Part 1 *General overview*
   - Part 2 *Planning and management*
   - Part 3 *Process for Developers*
   - Part 4 *Process for Acquirers*
   - Part 5 *Process for Evaluators*
   - Part 6 *Documentation of evaluation modules*

5. **ISO WD 9126 *Software quality characteristics*** — **ISO WD 9126 Software quality characteristics**
   - Part 1 *Quality characteristics and sub-characteristics*
   - Part 2 *External metrics*
   - Part 3 *Internal metrics*

6. **DIN 66234 *VDU Workstations*** — **DIN 66234 VDU Workstations**
   - Part 1 *Character shapes*
   - Part *2* *Perceptibility of characters*
   - Part 3 *Grouping and formatting of data*
   - Part 4 *not used*
   - Part 5 *Coding of information*
   - Part 6 *Design of workstation*
   - Part 7 *Design of work* environments
   - Part 8 *Design of dialogue design*
   - Part 9 *Measurement techniques*

- Part 10 *Minimum fact sheets*

**European Display Screen Directive**

7. ***European Display Screen Directive***
   Befaßt sich primär mit den physikalischen Anforderungen an die Arbeitsumgebung und Arbeitsbedingungen; enthält aber auch folgende Anforderungen, die alle Arbeitsstationen ab dem Jahre 1997 erfüllen müssen:
   - Software muß der Aufgabe angemessen sein.
   - Software muß einfach zu benutzen sein und, wo geeignet, an die Erfahrung oder das Wissen der BenutzerInnen anzupassen.
   - Systeme müssen Informationen in einem Format und einer Geschwindigkeit anzeigen, die an die BenutzerInnen angepaßt sind.
   - Die Prinzipien der Software-Ergonomie müssen angewendet werden.

### 2.2.5 Einschätzung der Leistungsfähigkeit von Standards

**Einwände gegen Standardisierungen**

Mit der Standardisierung hat sich ein Verfahren etabliert, daß auf alle Fragen zu einem bestimmten Themengebiet eine passende Antwort und Lösung angeboten werden müßte. Es bleibt aber die Frage, ob Standards dies erreichen können oder überhaupt sollen. Speziell im Bereich der Informationstechnologie sind diese Ansprüche umstritten. Die folgenden vier Fragen beschreiben die Einwände, die als Diskussionsanregung dienen sollen (vgl. HOLDAWAY & BEVAN S. 5ff, BEIMEL, SCHINDLER & WANDKE 1993 S. 134, NIELSEN F. 1996):

**Unbekannte Grundsätze**

1. **Sind die Grundsätze ausreichend bekannt?**
   Die bisherige Forschung ist noch nicht in der Lage, alle kognitiven Prozesse mit der Leistung individueller BenutzerInnen in Verbindung zu bringen, um strenge und präzise Standards zu bilden. Dies gilt insbesondere für Oberflächen, die auf Einflüsse unterschiedlicher Kulturen, Technologien, Aufgaben und Umgebungen reagieren müssen.

**Fortschrittshemmer**

2. **Hemmen Standards den Fortschritt?**
   Restriktive Standardisierungen können den Fortschritt hemmen, wenn deren Gestaltungsgegenstand nicht grundlegend erforscht wurde. Sind falsche Rahmen gesetzt worden, besteht die Gefahr, Kreativität und Innovation zu hemmen, die für einen konstruktiven und schöpferischen Fortschritt notwendig sind.

**Keine Garantie für gute Software**

3. **Garantieren Standards perfekte Software?**
   Ein umfassendes Verständnis für die Anforderungen der Usability erfordert das Verstehen der kognitiven Prozesse, die es BenutzerInnen erlauben, ihre Ziele effektiv, effizient und komfortabel zu erreichen. Da diese Prozesse in ihrer Vollkommenheit nicht verstanden werden, kann man die Dimensionen der Usability auch (noch) nicht überblikken.

   Es bleibt die Frage, ob es mit einer Standardisierung überhaupt möglich ist, daß zwei verschiedene DesignerInnen die aufgestellten Prinzipien gleich interpretieren und unabhängig voneinander konsistente Oberflächen gestalten.

4. **Führen Standards zwingend zu meßbaren Vorteilen?** **Keine meßbaren Vorteile**
   Die Standardisierung darf nicht nur um des Standards willen erfolgen. Es muß ein meßbarer Vorteil herauskommen, ansonsten besteht die Gefahr, daß die Standard-Oberflächen rigide und nicht an die verschiedenen Bedürfnisse der BenutzerInnen angepaßt werden.

### 2.2.6 Subjektive Bewertung der Standards

**Experten-Einschätzung zu DIN 66234 und ISO 9241 Part 11**

Um einen Überblick über die allgemeine Akzeptanz der vorgestellten Standards zu erhalten, wurde 1993 eine Befragung von Software-Ergonomie-ExpertInnen (n=90) aus aller Welt durchgeführt. In den Fragebogen wurden Bewertungen des DIN 66234 und der darauf basierenden ISO 9241 gefordert. Weiterhin sollten die ExpertInnen die allgemeinen Dialogprinzipien der ISO 9211 Part 11 beurteilen. Folgende Ergebnisse wurden erzielt (vgl. BEIMEL, SCHINDLER & WANDKE 1993 S. 135ff):

1. **Reichen die vorliegenden SW-Kenntnisse aus, um einen Standard zu entwickeln?** **Ausreichende Grundkenntnisse**
   Hierbei bejahten 50,7 % diese Frage, während 20,2 % unentschieden, und 21,3 % dies verneinten.

2. **Welchen Einfluß haben diese Standards auf die technische Weiterentwicklung von Dialogsystemen?** **Einflüsse auf den Fortschritt**
   65,9 % sind der Meinung, daß dieser Standard die Entwicklung fördere; 28,4 % glauben an keine und 5,7 % an negative Einflüsse.

3. **Sind die theoretischen Grundlagen dieses Standards bekannt?** **Kenntnisse der theoretischen Grundlagen**
   75,9 % der Befragten kannten die theoretischen Grundlagen dieses Standards, 73,6 % von ihnen befürworteten sie.

4. **Sind die allgemeinen Gestaltungsprinzipien bekannt?** **Kenntnisse der allgemeinen Gestaltungsprinzipien**
   Der übergroßen Mehrheit der Befragten sind die formulierten Gestaltungsprinzipien bekannt (90 % - 97 %). Hierbei sind nationale Unterschiede feststellbar:
   In Australien sind die Prinzipien der Aufgabenangemessenheit, Fehlerrobustheit und Individualisierbarkeit weniger bekannt als in Europa und Japan. **Nationale Unterschiede**

Insgesamt ist feststellbar, daß die aufgestellten Standards bei den Ergonomie-ExpertInnen be- und anerkannt sind.

### 2.2.7 Usability und Designaktivitäten

**Berücksichtigung von Usability im Entwicklungsprozeß**

Die Eigenschaften der Usability müssen während des Entwicklungsprozesses des Produktes berücksichtigt werden. Die nachfolgenden Schritte bieten einen allgemeinen, benutzerInnen-orientierten Ansatz, mit dem Usability während des Entwicklungsprozesses berücksichtigt werden kann (vgl. BEVAN S. 6):

1. Verstehe und spezifiziere den Gebrauchskontext (zukünftige BenutzerInnen, Aufgaben, Umgebung).
2. Spezifiziere die benutzerInnen- und organisationsbezogenen Ansprüche.

3. Stelle Designlösungen auf (Berücksichtigung allgemein gültiger Normen und allgemeinen Wissens, Prüfung und Evaluation dieser Lösungen).
4. Evaluiere das Design gegen die Ansprüche.

Dieser Prozeß muß iterativ im gesamten Entwicklungsprozeß durchgeführt werden. Es gibt ähnliche Methoden, die weit ausführlicher und umfassender in ihrer Ausprägung sind (vgl. NIELSEN 1993 S. 72ff). Verschiedene europäische Projekte (MAPI, INUSE, RESPECT) verfolgen diesen Ansatz seit Mitte der neunziger Jahre und erstellen Software-Projekte gemäß diesen Anforderungen (vgl. BEVAN 1996 S. 5f).

## 2.3 Evaluation

**Definition Evaluation**

Allgemein kann gesagt werden, daß Evaluation etwas mit „Bewerten" zu tun hat. Genauer ist Evaluation das „Sammeln und Kombinieren von Daten mit einem gewichteten Satz von Skalen, mit denen entweder vergleichende oder numerische Beurteilungen erlangt werden sollen." (WOTTAWA 1986 S. 707). Nicht immer werden bei Evaluationen feste Skalen verwendet, auch liegen Gewichtungen und Bewertungskriterien nicht unbedingt explizit vor. Im wissenschaftlichen Kontext wird Evaluation jedoch dahingehend verstanden, daß es sich bei den Beurteilungen nicht um rein subjektive Eindrücke handelt, sondern um systematisch gewonnene Erkenntnisse (vgl. WOTTAWA 1986 S. 707).

### 2.3.1 Der Evaluationsprozeß

**Gemeinsamkeiten von verschiedenen Evaluationen**

Alle Evaluationen, gleichgültig in welchem Zusammenhang sie durchgeführt werden, haben bestimmte Gemeinsamkeiten (vgl. KARAT J. 1997 S. 698f):

- Ausgangspunkt einer Evaluation ist der Untersuchungsgegenstand, das Objekt, das untersucht werden soll. Das Objekt muß nicht unbedingt gegenständlich sein: In der Pädagogik wird Evaluation mit „Bewerten von Handlungsalternativen" in Zusammenhang gebracht (vgl. WOTTAWA 1986 S. 707).
- Das Objekt soll bestimmte Eigenschaften (Attribute) besitzen. Die zukünftigen Eigenschaften müssen ermittelt und formuliert werden.
- In einem Prozeß werden tatsächliche mit den zuvor formulierten Eigenschaften des Objekts verglichen.

**Klassifizierung von Evaluationszielen**

Das Ziel einer Evaluation ist immer eine Bewertung „gegen etwas" - ganz gleich, ob dies nun implizit oder explizit benannt wird. HOLZ AUF DER HEIDE klassifiziert die Ziele einer Evaluation wie folgt (1993 S. 160):

- **Vergleichende Evaluation** (*Which is better?*): Es geht um einen direkten Vergleich zwischen alternativen Dialogsystemen.
- **Bewertende Evaluation** (*How good?*): Hier wird eine bestimmte gewünschte oder geforderte Systemeigenschaft geprüft.
- **Analysierende Evaluation** (*Why bad?*): Bei dieser Form der Evaluation geht es darum, Hinweise auf Schwachstellen zu bekommen, um direkt Gestaltungsvorschläge zu liefern.

**Teilschritte des Evaluationsprozesses**

Abgeleitet von diesen Gemeinsamkeiten setzt sich der Evaluationsprozeß mindestens aus den folgenden Teilschritten zusammen (vgl. GÖRNER & ILG 1993 S. 193ff):

- **Festlegung der erwarteten Eigenschaften des Zielprodukts.**
  Die Eigenschaften können entweder als absolute Werte oder relativ zu einem Vergleichsobjekt angegeben werden. Die zu erwartenden Eigenschaften sollen möglichst genau angegeben werden, damit auch vernünftige Metriken entwickelt werden können.[12]

- **Messung der tatsächlichen Eigenschaften des Zielprodukts.**
  In diesem Prozeß werden Daten irgendeiner Form erhoben. Die Daten können sowohl qualitativer als auch quantitativer Natur sein.

- **Interpretation der Ergebnisse und Abschlußbewertung.**
  Nachdem die Benutzungseigenschaften festgelegt wurden und eine Datenerhebung stattgefunden hat, ist ein Vergleich der erwarteten und tatsächlichen Eigenschaften und eine Interpretation der Ergebnisse nötig.

Je nach besonderer Ausprägung der Evaluation werden diese Teilschritte verfeinert. WIXON & WILSON (1997 S. 654) schlagen im Bereich des Usability-Engineering folgende Teilschritte in Form eines Algorithmus vor:

1. *Define measurable usability attributes. Set the quantitative levels of desired usability for each attribute. Together, an attribute and a desired level constitute a usability goal.*
2. *Test the product against the usability goals. If you meet your goals, no further design is needed.*
3. *If further design work is needed, analyze the problems that emerge.*
4. *Analyze the impact of possible design solutions.*
5. *Incorporate user-derived feedback in product design.*
6. *Return to Step 2 to repeat the test, analysis, and design cycle.*"

**Verschiedene Arten der Datengewinnung**

Jede Evaluation gewinnt im zweiten Teilschritt des Evaluationsprozesses Daten. GÖRNER & ILG gehen davon aus, daß sich die Art der Datengewinnung „auf ein Kontinuum zwischen subjektiven und objektiven Verfahren abbilden läßt." (1993 S. 195). Subjektive Evaluationen stützen sich auf die Erfahrung von Menschen. Objektive Evaluationsmethoden haben das Messen als Basis und sind unabhängig von der menschlichen Erfahrung. Diese Evaluationsmethoden sollten auch bei mehrfacher Anwendung immer zu den gleichen Ergebnissen führen. Subjektive Evaluationsmethoden können sowohl zu qualitativen Daten (z. B. Interview-Mitschnitte) als auch zu quantitativen Daten (Fragebogen mit *ranking*) führen, während objektive Methoden immer quantitatives Datenmaterial als Ergebnis haben.

**Verschiedene Techniken**

KARAT C.-M. (1994) kommt zu den gleichen Ergebnissen. Sie klassifiziert jedoch nicht nach der Art der Datengewinnung, sondern analysiert die verschiedenen Techniken.[13]

---

[12] Vgl. dazu auch die Ausführungen in Kapitel 2.3.3 zum quantitativen Evaluationsansatz.

[13] *is currently composed of two types of techniques ... (1) empirical usability testing in laboratary or field settings; and (2) a variety of usability inspection methods.* (KARAT C.-M. 1994 S. 203).

### 2.3.2 Evaluation von Software in bezug auf Usability

**Notwendigkeit von Evaluationen**

Molich und Nielsen demonstrierten bereits Ende der achtziger Jahre, wie schwierig software-ergonomisches Design bereits für ein kleines Anwendungsprogramm sein kann. Sie entwarfen eine Anwendung, bei der die BenutzerIn eine Telefonnummer eingeben konnte und als Rückmeldung Name und Adresse der zugehörigen Person bekam. Trotz weniger Interaktionsmöglichkeiten waren 31 Schwachstellen auf einer Bildschirmmaske enthalten. Molich und Nielsen starteten einen Wettbewerb im Internet: Die Teilnehmenden am Wettbewerb sollten so viel Schwachstellen wie möglich finden und Lösungsvorschläge zu gefundenen Mängeln vorlegen. Bei 77 Einsendungen wurden durchschnittlich nur 37 % der Fehler gefunden (vgl. GÖRNER & ILG 1993 S. 189).

**Die Rolle des User-Interface allgemein**

Die Gestaltung der Benutzungsoberfläche nimmt über die Hälfte (vgl. MAAß 1995 S. 231) des Software-Design-Prozesses in Anspruch. So wurden bereits 1992 (vgl. MYERS & ROSSEN):

48 % des Codes,
45 % der Zeit in der Design-Phase,
50 % der Zeit in der Implementierungsphase und
37 % der Zeit in der Wartungsphase

für die Anteile der Benutzungsoberfläche verwendet (n=74).

**Benutzbarkeit (usability) von Programmen**

Die Qualität eines benutzbaren Systems läßt sich daran messen, ob BenutzerInnen ihre Ziele mit dem Softwaresystem erreichen oder nicht bzw. wie effektiv sie diese Ziele erreichen (vgl. KARAT J. 1997 S. 693). Der Begriff Usability geht jedoch, wie wir in Kapitel 2.2 verdeutlicht haben, weiter: Ein benutzungsfreundliches System zeichnet sich nicht nur dadurch aus, daß ein Ziel erreicht, sondern daß es auf eine bestimmte Art und Weise erreicht wird. Hier wird also die Frage der Effizienz von Software gestellt.

Wenn wir vorhin davon gesprochen haben, daß wir bei einer Evaluation Objekte auf bestimmte Eigenschaften untersuchen, dann kann Usability allein nicht als eine solche Eigenschaft verstanden werden. Erfahrene BenutzerInnen können z. B. Programme nur über *shortcuts* bedienen, für sie sind Menüs überflüssig. Andere BenutzerInnen sind es gewohnt, alle Aktivitäten über Menüs zu steuern. Letztendlich stellt sich die Frage nach der Benutzbarkeit von Software immer in der Interaktion zwischen Software und Mensch in einem bestimmten Arbeitskontext bzw. einem Kontext der Benutzung[14] allgemein (vgl. KARAT J. 1997 S. 692).

**Definition Benutzungsprobleme (usability problems)**

Im Zentrum der Evaluation der Benutzungsoberfläche steht die Beseitigung von Benutzungsproblemen(*usability problems*). Wir schließen uns in der Definition von Benutzungsproblemen MACK & NIELSEN an: „*Usability problems can be defined as aspects of a user interface that may cause the resulting system to have reduced usability for the end user.*“ (1994 S. 3).

**Benutzungsprobleme vs. software-ergonomische Mängel**

Die Menge der Benutzungsprobleme und der software-ergonomischen Mängel muß nicht identisch sein. Manchmal entdecken BenutzerInnen Fehler gar nicht, die später Schäden hervorrufen können, z. B. Farbkombinationen, die subjektiv als optimal angesehen werden und objektiv (langfristig) zu Proble-

[14] *context of use.*

men führen können. Es scheint also kein Benutzungsproblem vorzuliegen jedoch ein software-ergonomischer Mangel.

### 2.3.3 Unterschiedliche Wege der Evaluation der Benutzungsoberfläche

**Vier Wege der Evaluation nach Nielsen und Mack**

MACK & NIELSEN (1994) nennen vier Wege, um eine Benutzungsoberfläche zu evaluieren:

1. formal über verschiedene Analysetechniken,
2. automatisch mit Hilfe einer Computerfunktion,
3. empirisch durch Experimente mit realen BenutzerInnen und
4. heuristisch durch einfaches Ansehen der Benutzungsoberfläche und deren Beurteilung durch Leitsätze (vgl. auch NIELSEN & MOLICH. 1990 S. 249).

**Quantitativer Evaluationsansatz**

Der quantitative Evaluationsansatz geht davon aus, daß Benutzbarkeit meßbar ist. Hier geht es um ein ingenieurmäßiges Herangehen an Softwarequalität. Dieses Herangehen wird mit dem Begriff des Usability-Engineering verbunden. Der angestrebte Grad von Benutzbarkeit soll von vornherein spezifiziert werden (vgl. MAAß 1993)[15]. Die entscheidende Tätigkeit ist hierbei das Messen: Jedem Attribut wird ein meßbares Kriterium zugeordnet, z. B. Aufgabenbearbeitungszeit, Häufigkeit des Gebrauchs von Handbüchern und Online-Hilfe, Anzahl Fehler pro Zeiteinheit. Bei der Systementwicklung wird versucht, das System nicht nur effizient, funktional korrekt und kostengünstig, sondern auch benutzbar im definierten Sinne zu gestalten.

**Objektivierung von subjektiven Zielen**

AnhängerInnen des quantitativen Ansatzes von Evaluation sind sehr darauf bedacht, möglichst viele meßbare Kriterien zu finden (vgl. WIXON & WILSON 1997 S. 656). Sie meinen, daß allgemeine Kriterien wie „leicht zu lernen“ oder „leicht zu benutzen“ nicht genügend definiert sind und keine Rolle im Software-Design-Prozeß spielen können. WIXON & WILSON (1997) schlagen vor, subjektive Evaluationsziele zu objektivieren. Ein subjektives Ziel sei: Das E-Mail-Programm soll leicht zu erlernen sein. Das Ziel kann dann wie folgt objektiviert werden: Neue BenutzerInnen sollen nach 30 Minuten Benutzung des Programms E-Mails lesen, senden und drucken können. Das zweite Ziel ist meßbar, das erste Ziel nur qualitativ zu bewerten. Wir werden im Kapitel 2.4 auf Methoden quantitativer Evaluation näher eingehen.

**Qualitativer Evaluationsansatz**

Subjektive Evaluationsmethoden liefern, wenn sie nicht mit Zufriedenheitsskalen arbeiten, qualitative Daten, die nur empirisch ausgewertet werden können. Diese Evaluationsmethoden (z. B. Lautes Denken) stützen sich im Bereich des Usability-Engineering auf Meinungen von BenutzerInnen ab. Die BenutzerInnen erkennen dabei Abweichungen vom und Störungen beim normalen Arbeitsablauf und benennen diese, ohne über ein spezifisches soft-

[15] MAAß (1993 S. 200) verbindet mit dem amerikanischen Begriff des Usability-Engineering den eher quantitativen, objektiven, ingenieurmäßigen Ansatz bei der Evaluation, während z. B. PFLÜGER (1992 S. 65) alle Maßnahmen in dem Oberbegriff Usability-Engineering zusammenfaßt und die eher subjektiven Maßnahmen als „Untermenge“ bezeichnet. Wir wollen im folgenden den Begriff Usability-Engineering im Sinne von PFLÜGER benutzen.

ware-ergonomisches Fachwissen zu verfügen (vgl. OPPERMANN & REITERER 1994).

**Partizipative, iterative Vorgehensweise**

Dieser Ansatz geht von einer partizipativen, iterativen Vorgehensweise aus. In jedem Stadium werden Tests gemacht, die qualitative Daten liefern. Außerdem geht es nicht so sehr darum, durch klare Kriterien Probleme im Design aufzudecken, sondern die Erfahrungen der ExpertIn bei der Beobachtung von BenutzerInnen auszunutzen. (vgl. MAAß 1993 S. 201). Solche Methoden sind z. B. Lautes Denken und Videoaufnahmen. Diesen Aspekt der Evaluation werden wir im Kapitel 2.5 näher beleuchten.

### 2.3.4 Zeitpunkt und Kosten von Evaluationsmaßnahmen

**Frühe Phasen der Softwareentwicklung**

Viele Studien haben ergeben, daß Evaluation in den frühen Phasen der Softwareentwicklung einsetzen muß (vgl. KAMINSKY 1992). Je früher Fehler im Design der Benutzungsoberfläche gefunden werden, desto größer ist die Chance und Motivation, den Fehler zu korrigieren (vgl. GÖRNER & ILG 1993 S. 190).

**Kosten von Fehlern**

Die Kosten für die Entdeckung und Beseitigung von Fehlern steigen exponentiell mit der Dauer zwischen dem Entstehen und der Entdeckung des Fehlers. Ein Fehler in der Anforderungsdefinition z. B. kostet ca. 30mal mehr, wenn ihn die EndbenutzerInnen entdecken, als wenn dieser Fehler beim Review der Anforderungsdefinition entdeckt worden wäre (vgl. PFLÜGER 1992 S. 21, TRAUBOTH 1993 S. 74):

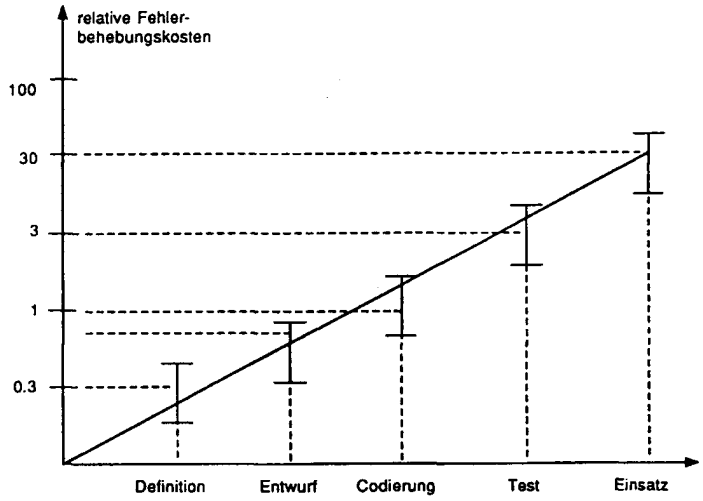



Abbildung 2: Relative Fehlerkostenbehebung (PFLÜGER 1992 S. 21)

**Ökonomische Vorteile liegen nicht auf der Hand**

Trotz dieser Erkenntnis und der Tatsache, daß die Produktionsverluste, die durch schlecht gestaltete Benutzungsoberflächen entstehen, immens sind (vgl. MAAß 1995 S. 222), hat die Software-Ergonomie sehr stark damit zu kämpfen, daß die ökonomischen Vorteile einer benutzerInnengerechten Systemgestaltung nicht offen auf der Hand liegen (vgl. MAAß 1995 S. 223).

### 2.3.5 Summative versus formative Evaluation

**Ganzheitlichkeit von Evaluationen: Summative Evalutation**

Softwareentwicklung im weiteren Sinne gesehen ist nicht nur die Gestaltung des Produkts, sondern auch die Gestaltung des Umfelds. Die Vorgehensweise der Gestaltung ist zunächst die Bearbeitung des organisatorischen Bereichs[16]. Hier geht es darum, die organisatorische Arbeitsgestaltung wie Arbeitszeit, Arbeitsablauf und Arbeitsbeanspruchung optimal zu gestalten (vgl. MAAß 1993, OPPERMANN & REITERER 1994, OPPERMANN et al. 1992). Nach dieser Phase wird dann die Software und zum Schluß die daraus resultierende Hardware gestaltet.

**Spezifische Richtung von Evaluationen**

Analog zur Vorgehensweise in Abbildung 3 folgt aus dieser ganzheitlichen Betrachtungsweise auch eine spezifische Richtung der Evaluation (vgl. OPPERMANN & REITERER 1994). OPPERMANN et al. fordern: „Evaluationsverfahren sollten alle relevanten ergonomischen Eigenschaften von Dialogsystemen ... in einer ganzheitlichen Weise erfassen." (1992 S. 5).

**Evaluation in der Praxis**

Diesem Gestaltungs- und Bewertungsansatz steht die Praxis gegenüber. WIXON & WILSON meinen dazu polemisch: „*Bluntly stated, while science is concerned with uncovering thruth, regardless of time or cost, engineering is concerned with the art of the possible within constrainded resources.*" (1997 S. 655).

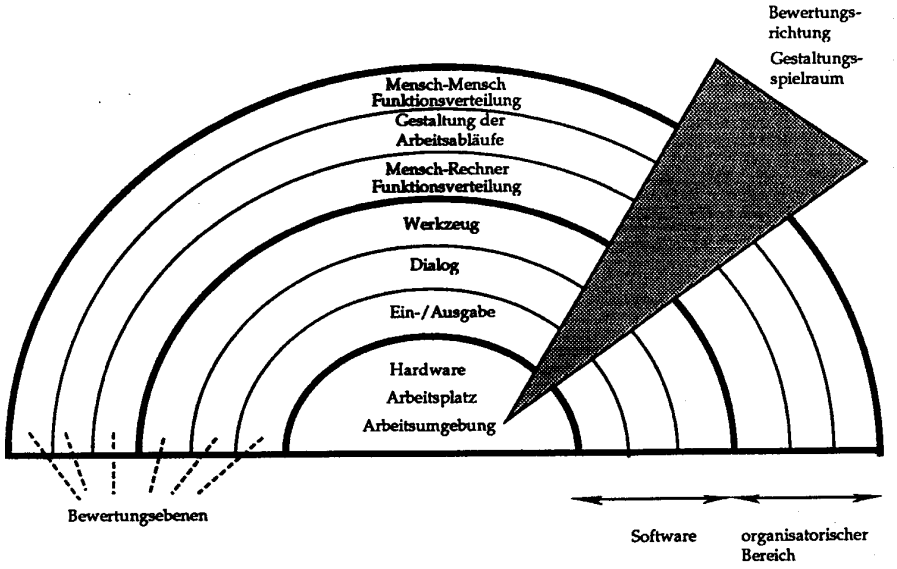



Abbildung 3: Richtungen und Ebenen der Evaluation (OPPERMANN & REITERER 1994 S. 339)

**Spannungsfeld zwischen Wissenschaft und Praxis**

Reale Evaluationstechniken stehen in diesem Spannungsfeld zwischen Wissenschaft und Praxis. In der Praxis stehen kleine Budgets Ansprüchen von vollständigen Evaluationen gegenüber. Im Hinblick auf den Zustand, daß es ein großes Defizit bei der Anwendung von Evaluationsmaßnahmen im Bereich der Benutzungsoberflächen gibt, ruft Nielsen allen zu: „*any data is data*" und „*anything is better then nothing*" (NIELSEN S. 15).

**Geeignete Evaluationsverfahren**

Um dieses Spannungsfeld aufzulösen, plädiert HOLZ VON DER HEIDE für ein geeignetes Evaluationsverfahren: „Um wirklich praxisrelevant zu sein, sollte

[16] OPPERMANN et al. nennen diesen Gestaltungsbereich Organisations-Ergonomie (1992).

die Erhebung der Daten und deren Auswertung nur einen relativ geringen finanziellen, zeitlichen und technischen Aufwand erfordern (1993 S. 159). Er plädiert weiter für Evaluationsverfahren, die universell einsetzbar und für Dialogsysteme unterschiedlicher Funktionalität und Komplexität anwendbar sind.

**Evaluation von Teilbereichen - formative Evaluation**

Die meisten Evaluationsmethoden heben einen bestimmten Aspekt der Software-Ergonomie hervor und bieten Hilfsmittel, diesen Aspekt zu evaluieren (z. B. die Explorationsfreundlichkeit der Software). Diese Art der Evaluation wird auch formative Evaluation genannt (vgl. HOLZ AUF DER HEIDE 1993 S. 160). Alternativ dazu spricht man von summativer Evaluation, bei der das Dialogsystem in seiner Gesamtwirkung im Vordergrund steht.

**Konzentration auf bestimme Evaluationsmethoden**

Im weiteren Verlauf der Arbeit werden wir im wesentlichen die Evaluationsmethoden betrachten, die während des gesamten Software-Entwicklungszyklus eingesetzt werden können und sowohl effektiv als auch effizient sind. Im Hinblick auf eine Software, die partielle Tests unterstützt, erscheinen uns die ganzheitlichen Methoden nicht geeignet.

**Plädoyer für einen Methodenmix**

Es gibt nicht die perfekte Evaluationsmethode. Jede Methode hat ihre spezifischen Vor- und Nachteile. In der Praxis wird es deshalb besser sein, verschiedene Evaluationsmethoden bzw. einen Methodenmix anzuwenden. (vgl. OPPERMANN & REITERER 1994, ANSORGE & HAUPT 1997).

**Kriterien für die Wahl einer spezifischen Evaluationsmethode**

Für die Wahl einer spezifischen Methode müssen folgende Fragen beantwortet werden (vgl. KARAT J. 1997 S. 693, WOTTAWA 1986 S. 708):

- **Was ist das Ziel der Evaluation?**
  Geht es um einen abschließenden Test des Gesamtsystems, oder sollen bestimmte Aspekte überprüft werden?
- **Wer soll die Evaluation durchführen?**
  Ist es besser, eine ExpertIn das System inspizieren zu lassen, oder sollen reale BenutzerInnen die Evaluation durchführen?
- **Wo soll die Evaluation durchgeführt werden?**
  Soll sie in einer künstlich geschaffenen Einrichtung (Labor) oder z. B. am Arbeitsplatz durchgeführt werden.
- **Welche Informationen werden gesammelt?**
  Was soll gemessen werden, welche Daten sind nützlich?
- **Wer sind die Adressaten des Tests?**
  Inwieweit soll der Test beispielsweise das weitere Handeln der EntwicklerInnen beeinflussen?
- **Wie groß sind die Ressourcen?**
  Wieviel Geld und Zeit steht für die Evaluation zur Verfügung?

## 2.4 Software-ergonomische Reviews

**Definition von software-ergonomischem Review**

Software-ergonomisches Review ist der allgemeine Begriff für eine Reihe von quantitativen Evaluationsmethoden, bei der EvaluatorInnen mit einer bestimmten Art von ExpertenInnenwissen relevante Teile der Benutzungsoberfläche untersuchen und prüfen (vgl. MACK & NIELSEN 1994 S. 1f).

In der IEEE-Norm 729-1983 wird das Review definiert als ein „mehr oder weniger formal geplanter und strukturierter Analyse- und Bewertungsprozeß, in dem Projektergebnisse einem Team von Gutachtern präsentiert und von diesen kommentiert oder genehmigt werden." (WALLMÜLLER 1990 S. 146).

**Verschiedene Begrifflichkeiten**

Bei der Beschreibung der verschiedenen Review-Methoden gibt es in der deutschen Literatur keine eindeutige Einigung in der Wahl der Begriffe. WALLMÜLLER schreibt beispielsweise., daß der Unterschied zwischen Walkthroughs und Inpektionen der ist, „daß Inspektionen formaler geplant und durchgeführt werden als Walkthroughs" (1990 S. 152), während z. B. TRAUBOTH Inspektionen als „weitgehend informale, manuelle Prüfungen und Untersuchungen" definiert (1993 S. 150). Auch in der englischen Literatur gibt es ähnliche Differenzierungen (vgl. KNIGHT & MYERS 1993, MACK & NIELSEN 1994).

**Usability inspection methods**

Quantitative Evaluationsmethoden gibt es in unterschiedlicher Ausprägung. Wir wollen im folgenden auf eine besondere Klasse von Review-Methoden eingehen, die auf den Tagungen des CHI 1990 und 1992 und in der dazugehörigen Literatur diskutiert und zusammenfassend *usability inspection methods* genannt wurden (vgl. NIELSEN & MOLICH 1990, LEWIS et al. 1990, KARAT C.-M., CAMPBELL & FIEGEL 1992, NIELSEN 1992, MACK & NIELSEN 1994, KAHN & PRAIL 1994).[17]

### 2.4.1 Usability inspection Methods

**Usability inspection**

*Usability inspection*[18] ist der Oberbegriff für eine Reihe von software-ergonomischen Qualitätssicherungsmaßnahmen, die seit der CHI-Konferenz 1990 in Seattle vermehrt diskutiert und eingesetzt werden (vgl. MACK & NIELSEN 1994, VIRZI 1997).

**Zusammenfassung verschiedener Methoden**

NIELSEN (1995) faßte diese Methoden auf der CHI Konferenz 1995 zusammen:

- ***Heuristic evaluation*:**[19]: Dies ist die am wenigsten formale Methode. Die Benutzungsoberfläche wird anhand anerkannter Leitsätze (Heuristiken) von mehreren EvaluatorInnen daraufhin untersucht, ob diese Leitsätze eingehalten werden.
- ***Cognitive walkthroughs*:** Beinhaltet eine sehr detaillierte Struktur, mit der das Review vorgenommen wird. Diese Methode untersucht in erster Linie, wie explorationsfreundlich eine Benutzungsoberfläche ist.

[17] Wir werden im folgenden vornehmlich die englischen Begriffe benutzen.

[18] Wir benutzen diesen Begriff synonym zu software-ergonomisches Review.

[19] In Anbetracht der Tatsache, daß dieses Wort im Deutsceh angeglichen werden kann, benutzen wir nachfolgend den Begriff Heuristische Evaluation.

- ***Formal usability inspections*:** Benutzt eine Prozedur in sechs Schritten. Diese Methode kombiniert die Heuristische Evaluation mit dem *cognitive walkthrough*.
- ***Pluralistic inspection*:** Bei dieser Methode werden Szenarios verwendet. BenutzerInnen, ExpertInnen und EntwicklerInnen gehen durch das Szenario und diskutieren alle Dialogelemente.
- ***Feature inspection*:** Hier werden typische Arbeitsaufgaben verwendet, um bestimmte Fehler zu finden: zu lange Sequenzen, unhandliche Abfolgen von Dialogschritten, Schritte, die normale BenutzerInnen nie benutzen würden und Schritte, die ein erhebliches Wissen voraussetzen, um überhaupt erfolgreich zu sein.
- ***Consistency inspection*:** Wird überwiegend für große Projekte durchgeführt, um zu untersuchen, ob verschiedene Funktionen in allen Kontexten gleich ausgeführt werden können.
- ***Standards inspection***: Betrachtet, ob Vorschriften (z. B. Styleguides) richtig angewendet wurden.

### 2.4.2 Verschiedene Ausprägungen von Usability Inspection Methods

**Gemeinsamkeiten und Unterschiede**

Alle diese Methoden haben Gemeinsamkeiten und Unterschiede, die sich in folgenden Merkmalen niederschlagen:

1. **Die Expertise der ExpertInnen**
   Allen Methoden ist gemeinsam, daß sie von speziellen ExpertInnen (*usability experts*) geplant und durchgeführt werden. Reale BenutzerInnen des Systems werden bei diesen Methoden nicht berücksichtigt. Die speziellen Kenntnisse der ExpertInnen können unterschiedlich sein: Kenntnisse in Software-Ergonomie, Erfahrung im Umgang mit dem Anwendungsprogramm oder auch spezielles Wissen im Arbeitsbereich des Programms. Einige Methoden können in weniger als einer Stunde erlernt werden (z. B. Heuristische Evaluation), während andere eine sehr lange Erfahrung im Umgang mit der Evaluation von Benutzungsoberflächen benötigen.
2. **Die Anzahl der Evaluatoren**
   Einige Inspektionsmethoden werden in Gruppen durchgeführt, andere zwar mit mehreren ExpertInnen, jedoch individuell und nacheinander.
3. **Ganzheitlichkeit der Inspektion**
   Grundsätzlich sind diese Methoden keine ganzheitlichen Evaluationsmethoden. Die Methode des *cognitive walkthrough* beispielsweise untersucht die Benutzungsoberfläche auf deren Explorationsfreudigkeit. Andere Methoden gehen auf spezifisch andere Aspekte von Software-Ergonomie ein, unter anderem auf die Einhaltung von Styleguides.

KARAT, CAMPEL & FIEGEL (1992) fanden heraus, daß sich die Ergebnisse verschiedener Methoden nur zu etwa einem Drittel überschneiden.

**Cognitive Walkthrough und Heuristische Evaluation**

Wir werden im folgenden eine formale und eine mehr informale Methode der *usability inspection* vorstellen: Das *cognitive walkthrough* und die Heuristische Evaluation. Das *cognitive walkthrough* kann als beispielsweise herangezogen werden, weil es sich nicht wesentlich von anderen ähnlichen Verfahren wie die *structed walkthroughs* von Yourdon oder Reviews, *walkthroughs* und *inspections,* wie sie bei Weinberg und Freedman beschrieben werden, unterscheidet (vgl. LEWIS et al. 1990 S. 235). Die Heuristische Evaluation haben wir gewählt, weil sie eine eher informale Methode ist, die sehr schnell erlernbar ist und angewendet werden kann.

### 2.4.3 Formale Reviews am Beispiel des cognitive walkthrough[20]

**Wozu dient das cognitive walkthrough**

Das *cognitive walkthrough* wurde entwickelt, um Entwurfteams in die Lage zu versetzen, frühzeitig Fehler im Design der Benutzungsoberfläche zu finden und zu berichtigen. Ein Vorteil dieser Methode liegt darin, daß sie mit minimalem Training erlernbar ist. Mit Hilfe von Fragestellungen wird das Programm inspiziert: Positive Beantwortung der Fragen deutet darauf hin, daß die Benutzungsoberfläche leicht zu erlernen ist. Negative Antworten suggerieren, daß die untersuchte Prozedur schwer zu erlernen ist, und zeigen potentielle software-ergonomische Fehler auf (vgl. LEWIS et al. 1990).

**Die Ebene der BenutzerInnen**

Die Methode erfordert keinen funktionierenden Prototypen oder die Beteiligung von BenutzerInnen bei der Durchführung des Tests. Die Durchführenden des Walkthroughs begeben sich auf die Ebene der BenutzerInnen. Sie erkennen dadurch Probleme, die die BenutzerInnen im Umgang mit dem Programm haben (vgl. RIEMAN, FRANZKE & REDMILES 1995).

**Charakteristik des cognitive walkthrough**

Einer der Entwickler dieser Evaluationsmethode, Clayton Lewis, charakterisiert das *cognitive walkthrough* wie folgt: „*First, the method is performed by an analyst and reflects the analyst s judgements, rather then based on data from test users. That is, the CW is not a user test. Second, the CW examines specific user tasks, rather than assessing the character of an interface as a whole ... Third, the CW analyzes correct sequence will actually be followed by users.*“ (LEWIS & WHARTON 1997 S. 717).

**Welche anderen Methoden sind vergleichbar?**

Das *cognitive walkthrough* hat den gleichen Ablauf und die gleichen Ziele wie andere Typen von Walkthroughs so die *code-walkthroughs*. Es ist ein Prozeß, bei dem auf der einen Seite ein Produkt präsentiert wird und auf der anderen Seite eine oder mehrere GutachterInnen das Produkt nach bestimmten Kriterien inspizieren (vgl. WHARTON et al. 1994). Wie das *programming walkthrough* ist auch das *cognitive walkthrough* aufgabenorientiert (vgl. LEWIS & WHARTON 1997).

**Ursprung des cognitive walkthroughs**

Das *cognitive walkthrough* basiert auf Forschungen von Lewis und Polson. Das Modell kombiniert eine regelbasierte Umsetzung der kognitiven Theorien von Polson mit den Forschungen von Lewis über das explorative Lernen. Carroll, Rosson und Fischer haben festgestellt, daß die meisten BenutzerInnen eine bestimmte Vorgehensweise bei der Erforschung neuer Software bevorzugen, nämlich die Exploration. Dieses Herangehen kann als „Lernen durch schritt-

[20] Eine ausführliche Beschreibung von *cognitive walkthroughs* in ihrer Entwicklung von 1990 bis heute bietet LEWIS & WHARTON (1997) und John & Packer (1995).

weises Ausprobieren“ charakterisiert werden (vgl. WHARTON et al. 1994 S 105).

Die Autoren schlußfolgern aus ihren Forschungen, daß eine Software vor allem leicht erlernbar sein sollte. Andere software-ergonomische Aspekte wie Zweckmäßigkeit und leichte Benutzbarkeit korrelieren mit der leichten Erlernbarkeit von Software (vgl. WHARTON et al. 1994 S. 107).

**Das Modell des explorativen Lernens**

Das Modell des explorativen Lernens hat mehrere Phasen (vgl. LEWIS et al. 1990):

- Die Phase : Problem lösen (*problem-solving component*).
- Die Phase: Lernen (*learning component*).
- Die Phase: Ausführen (*execution component).*

**Problem lösen**

Die erste Phase des explorativen Lernens beruht auf folgender Beobachtung: BenutzerInnen haben bei einer Software, die explorationsfreundlich entwickelt wurde, mehrere Alternativen, ein spezifisches Ziel zu erreichen. Sie werden die Alternative wählen, die ihrem Kenntnisstand und ihren Erwartungen am ähnlichsten sind. Sie werden versuchen, die Aktion (Wahl eines Menüeintrags, Betätigen eines Schalters etc.) zu wählen, deren Beschriftung ihren gewünschten Zielen am nächsten kommt. Nachdem die BenutzerInnen ihre Aktionen durchgeführt haben, vergleichen sie die Reaktionen des Systems mit ihren Zielen. Falls die Reaktion des Systems nicht erwünscht ist, müssen sie wieder zurück in den vorherigen Zustand, um eine andere Alternative zu suchen.

**Lernphase**

Die Lernphase setzt dann ein, wenn die gewählte Aktion zum Ziel führt. Der Weg zum Ziel wird als eine Regel begriffen. Polson und Lewis haben erkannt, daß es eine größere Herausforderung ist, eine Problemlösungsstrategie zu finden, als die gewählte Strategie in Regeln für weitere Aktivitäten zu transformieren (vgl. LEWIS et al. 1990 S. 236).

**Ausführungsphase**

Die Ausführungsphase des Modells besagt, daß BenutzerInnen, wenn sie das erste Mal ein spezifisches Programm vor sich haben, zunächst eine ihrer gelernten Regeln anwenden wollen, um zu ihrem Ziel zu gelangen. Wenn diese Strategie nicht greift, werden sie wieder zur ersten Phase zurückkehren.

**Schlußfolgerungen für das Design von user interfaces**

Polson und Lewis haben aus dieser Theorie folgende Schlußfolgerungen für das Design von Benutzungsoberflächen gezogen (LEWIS et al. 1990 S. 236):

- „*Make the repertory of available action salient.*
- *Provide an obvious way to undo actions.*
- *Offer few alternatives.*
- *Require as few choices as possible*“.

**Durchführung des cognitive walkthrough**

Bei einer realistischen Arbeitsaufgabe werden die BenutzerInnen Zwischenziele definieren, um das Gesamtziel zu erreichen. Sie werden die drei Schritte des explorativen Lernens so lange wiederholen, bis das Gesamtziel erreicht ist. Hier setzt das *cognitive walkthrough* ein. Die Durchführung des Walkthroughs umfaßt zwei Phasen: Die Vorbereitungsphase und die Analysephase. In der

Vorbereitungsphase müssen folgende Schritte beobachtet werden (vgl. RIEMAN, FRANZKE & REDMILES 1995):

1. Eine allgemeine Beschreibung der potentiellen BenutzerInnen des Systems, deren Kenntnisstand und Erfahrung im Umgang mit Computer-Systemen.
2. Eine präzise Beschreibung einer oder mehrerer typischer Arbeitsaufgaben, die mit dem System erledigt werden sollen.
3. Eine Liste aller Möglichkeiten, wie die entsprechende Arbeitsaufgabe gelöst werden kann.

**Organisation durch EntwicklerInnen**

Vom Grundgedanken her sollte das Walkthrough von einzelnen EntwicklerInnen oder einer Gruppe von EntwicklerInnen des Systems durchgeführt werden. Es können auch nur einzelne Teile der Benutzungsoberfläche untersucht werden.

**Fragebogen des cognitive walkthrough**

Für jede Aktivität wird ein Protokoll angefertigt, das die nachfolgenden Fragen an jeden Arbeitsschritt beinhaltet (vgl. LEWIS et al. 1990).

Bei jeder dieser Fragen müssen die EvaluatorInnen entscheiden, ob keine, einige, mehr als die Hälfte oder die meisten BenutzerInnen Probleme haben werden.

1. Beschreibung des Ziels der BenutzerIn.
2. Nächste atomare Aktion, die die BenutzerIn ausführen kann.
   - 2a. Ist diese Aktion vorhanden?
   - 2b. Führt diese Aktion zum Ziel?
3. Wie bekommt die BenutzerIn eine Beschreibung für diese Aktion?
   - 3a. Gibt es Probleme beim Zugriff?
4. Wie verbindet die BenutzerIn die Beschreibung mit der Aktivität?
   - 4a. Gibt es Probleme beim Verbinden?
5. Sind alle anderen Aktionen weniger geeignet?
6. Wie führt die BenutzerIn die Aktion aus?
   - 6a. Gibt es Probleme?
7. Falls die Aktion länger dauert, wird die BenutzerIn vor der Aktion davon in Kenntnis gesetzt?
8. Die Aktion wird durchgeführt. Beschreibe die Antwort des Programms
   - 8a. Ist es offensichtlich, daß die Aktion zum Ziel geführt hat?
   - 8b. Kann die BenutzerIn auf wichtige Informationen zugreifen? Antwortet das System?
9. Beschreibung des veränderten Ziels, falls nötig.
   - 9a. Ist es offensichtlich, das das Ziel geändert werden muß?
   - 9b. Falls der Arbeitsschritt beendet ist, ist dies offensichtlich?

**Stärken und Schwächen des cognitive walkthrough**

In einer Studie fanden Lewis und Polsen heraus, daß mit Hilfe des *cognitive walkthroughs* ca. 50 % der Designprobleme gefunden werden können. Dies ist zwar nicht viel, aber die Autoren weisen darauf hin, daß diese Methode nicht sehr kostenbelastend ist: „*Our arguments are that the walkthrough with a very limited*

*investment in resources, approximately an hour per task per interface, can detect almost 50 percent of the problems encoutered by users of the design.*" (LEWIS et al. 1990 S. 241)[21].

Die Autoren argumentieren weiter, daß diese Methode sicherlich nicht andere ersetzt, jedoch schon sehr früh im Softwareentwicklungszyklus eingesetzt werden kann - im Prinzip nach der Spezifikationsphase. Das Walkthrough kann an Papier-Simulationen erprobt werden oder an sehr rudimentären Prototypen (vgl. WHARTON et al. 1992, WHARTON et al. 1994).

**Cognitive Jogthrough**

ROWLEY & DAVID (1992) meinen, daß die Prozedur des Walkthrough sehr zeitaufwendig ist. Sie testeten diese Methode für eine Inhouse-Lösung mit ExpertInnen, EntwicklerInnen und BenutzerInnen aus der eigenen Firma. Sie hatten in 90 Minuten erst zehn atomare Aktionen inspiziert: „*This ist roughly equivalent to opening a single file and displaying a windowful of application parameters to edit (a minor protion of the task subset we wanted to evaluate).*"[22]. Sie schlagen als Alternative das *jogthrough* vor. Beim *jogthrough* wird die Evaluationsprozedur technisch unterstützt: durch eine Videokamera, die mit einem *event-logger* verbunden ist. Die Kommentare für jeden einzelnen Schritt werden jetzt nicht mehr schriftlich protokolliert, sondern direkt aufgezeichnet. Durch die Kombination von Video und *event-logging* ist es schnell möglich, die Videosequenz abzuspielen, die zu einer spezifischen Aktion gehört (vgl. ROWLEY & DAVID 1992 S. 392ff). Neben dem Zeitfaktor gibt es einen zweiten, wichtigen Vorteil: Es werden nicht nur die vorgegebenen Fragen beantwortet und somit dokumentiert, sondern alle Kommentare, die zu einem Schritt gemacht wurden (vgl. LEWIS & WHARTON 1997 S. 724).

### 2.4.4 Informale Reviews am Beispiel Heuristische Evaluation[23]

**Heuristische Evaluation**

NIELSEN & MOLICH (1990) haben eine Methode der Evaluation vorgeschlagen, in der Benutzungsoberflächen anhand von bestimmten Merkmalen beurteilt werden. Diese Methode nennen sie Heuristische Evaluation. Die Heuristische Evaluation erfordert eine kleine Anzahl von ExpertInnen, die die Benutzungsoberfläche daraufhin untersuchen, ob anerkannte Richtlinen (die Heuristiken) eingehalten wurden. Das Problem ist, daß typische Styleguides und andere Richtlinien so viele Regeln haben (oft mehr als 1.000), daß auch Software-Ergonomie-ExpertInnen nicht alle überprüfen können und letztendlich nur auf ihre Erfahrung und Intuition angewiesen sind (vgl. NIELSEN & MOLICH 1990 S. 249). NIELSEN & MOLICH verminderten die Komplexität, indem sie die Regeln auf zehn reduzierten: „*For the discount method I advocate cutting the complexity by two orders of magnitude, to just 10 rules, relying on a small set of broader heuristics such as the basic usability principles.*" (NIELSEN 1993 S. 19).

**Ursprung und Theorie**

In der Praxis werden die meisten Evaluationen in verschiedenen Formen Heuristischer Evaluation durchgeführt. Das Problem ist jedoch, daß die meisten

[21] RIEMAN et al. (1991) reduzierten die Zeit für eine Aufgabe von einer Stunde auf 5 - 10 Minuten, indem sie ein System entwickelten, das das *cognitive walkthrough* durch Standardfragestellungen unterstützt.

[22] LEWIS et al. (1990) sprechen von einer Stunde, Task, Rowley von 10 atomaren Aktionen/90 Minuten. Auch wenn die Zeiteinteilungen nicht vergleichbar sind, wird deutlich, daß der Zeitaufwand dieser Methode enorm ist.

[23] Eine vollständige Beschreibung der Methode in NIELSEN (1993).

EvaluatorInnen, die sie anwandten, nichts Genaues über die Methode wußten und sie als minderwertig einstuften (vgl. NIELSEN & MOLICH 1990 S. 249). NIELSEN & MOLICH beschäftigten sich seit Ende der achtziger Jahre mit diesem Problem und haben bis heute verschiedene Studien durchgeführt, um die Wirkungsweise der Heuristischen Evaluation zu erläutern und ihre Effektivität zu beweisen.

**Durchführung der Heuristischen Evaluation**

In der Heuristischen Evaluation inspiziert jede EvaluatorIn für sich die Benutzungsoberfläche. Erst nach der Evaluation werden die Ergebnisse verglichen und die gefundenen Fehler diskutiert. Dies ist wichtig, damit die verschiedenen EvaluatorInnen unvoreingenommen in den Test gehen. Die Ergebnisse können in Berichten festgehalten oder einer BeobachterIn mündlich mitgeteilt werden. Geschriebene Berichte haben den Vorteil, daß sie eine formale Aufzeichnung des Tests sind, und den Nachteil, daß sie gelesen und ausgewertet werden müssen. Während des Tests geht die EvaluatorIn mehrere Male über die verschiedenen Dialoge und vergleicht sie mit einer Liste von anerkannten Prinzipien. Wichtig ist, daß die EvaluatorInnen so detailliert wie möglich die einzelnen Probleme auflisten. Es soll nicht zusammengefaßt werden: jedes Problem soll separat bewertet werden (vgl. NIELSEN 1994a).

**Die Heuristiken**

Die zehn Regeln, die Nielsen und Molich in die Heuristische Evaluation aufnahmen, sind (vgl. NIELSEN 1993 S. 20 und S. 115 ff):

1. *Simple and natural dialogue.*
2. *Speak the users language.*
3. *Minimize the users memory load.*
4. *Provide consistency.*
5. *Provide feedback.*
6. *Provice clearly marked exits.*
7. *Provide shortcuts.*
8. *Provide good error messages.*
9. *Prevent errors.*
10. *Provide help and documentation.*

**Eight golden rules of dialog design**

Bereits vor dieser Veröffentlichung gab es Studien, die auf ähnliche Regeln hinwiesen, wie z. B. die „*Eight golden rules of dialog design* von SHNEIDERMAN (1987 S. 60ff).

**Erweiterte Regeln**

Die einzelnen Regeln sind seither erweitert oder verändert worden. So schlugen MULLER et al. (1995) noch drei weitere Heuristiken vor:

11. *Respect the user and her/his skills.*
12. *Pleasurable experience with the system.*
13. *Support quality work.*

MULLER et al. (1995) wiesen nach, daß durch diese drei zusätzlichen Regeln 15 % mehr *usability problems* gefunden werden als ohne diese Regeln.

**Das Besondere an der Methode**

Die Heuristische Evaluation, wie sie NIELSEN & MOLICH (1990) vorschlagen, ist kein systematischer Weg, eine Benutzungsoberfläche zu beurteilen. NIELSEN (1993) weist aber nach: Wenn mehrere Personen die Evaluation auf diese Weise durchführen, werden über 90 % der Probleme aufgedeckt. NIELSEN & MOLICH schlagen drei bis fünf Beurteiler vor (1990). Diese Methode ist nur wirksam, wenn die EvaluatorInnen fundierte Kenntnisse über

die realen BenutzerInnen des Systems und die Arbeitsaufgabe haben. NIELSEN nennt diese beiden Elemente die kritischen Faktoren im Usability-Engineering (1992 S. 45). Deshalb sollte die Heuristische Evaluation mit empirischen Tests kombiniert werden. NIELSEN empfiehlt die Methode des Lauten Denkens. (1992).

**Wer findet welche Probleme?**

Ein Evaluator findet λ % aller Fehler in der Benutzungsoberfläche (*usability problems*). Jeder weitere Evaluator findet wiederum weitere λ % der übrig gebliebenen Fehler. Es entsteht so folgende Kurve für λ = 31 % (durchschnittlich):

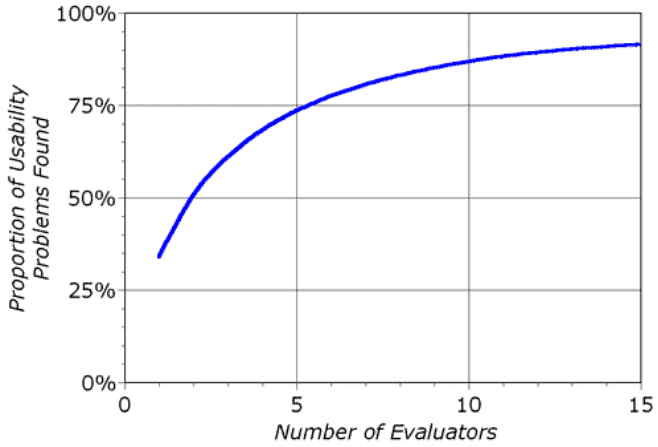



Abbildung 4: Verhältnis von Anzahl der Evaluatoren zu gefundenen Fehlern (NIELSEN & LANDAUER 1993)

**Verhältnis von EvaluatorInnen und Problemen**

λ ist bei verschiedenen Projekten unterschiedlich. NIELSEN & LANDAUER (1993) fanden heraus, daß es eine Formel für das Verhältnis von EvaluatorInnen und gefundenen Problemen in der Benutzungsoberfläche gibt:

$$Found(i) = N(1-(1-\lambda)^{i})$$

Dabei ist N die Gesamtanzahl der Fehler und i die Anzahl der EvaluatorInnen. Durch geeignete Umstellung der Formel läßt sich bei spezifischen Projekten λ wie folgt berechnen:

$$\lambda = 2-(Found(2)/Found(1))$$

**Zusammenfassung von Nielsen**

NIELSEN faßte die Ergebnisse der Heuristischen Evaluation 1992 zusammen. Dort weist er nach, daß bestimmte ExpertInnen mehr Fehler finden als andere. Er unterscheidet zwischen Studierenden[24], ExpertInnen und doppelt-qualifizierten ExpertInnen. Doppelt-qualifizierte ExpertInnen sind solche, die neben den Kenntnissen in Software-Ergonomie auch noch Kenntnisse in einzelnen Teilen der evaluierten Benutzungsschnittstelle haben. In der Studie läßt Nielsen eine Heuristische Evaluation durch ExpertInnen mit unterschiedlichen Kenntnissen durchführen. Software-Ergonomie-Expertinnen finden im Durchschnitt 74 %– 87 % der Probleme (bei drei bis fünf Expertinnen). Die doppelt-qualifizierten Expertinnen finden bei zwei bis drei EvaluatorInnen im

[24]Es handelte sich um Studierende, die gerade ihr erstes Programmierpraktikum hinter sich gebracht und noch keine Informationen über Software-Ergonomie hatten.

Schnitt 81 % – 90 % der Probleme, die Studierenden finden im Schnitt 22 % bis 29 % der Probleme und fünf Studierende finden etwas über die Hälfte der Probleme (vgl. NIELSEN 1992).

**Heuristische Evaluation braucht Ergonomie-ExpertInnen**

Die Heuristische Evaluation benötigt ExpertInnen, die viel Erfahrung im Bereich der Software-Ergonomie haben. In einer Feldstudie mit vier Untersuchungen fand eine EvaluatorIn 51 %, 38 %, 26 % und 20 % der Fehler (NIELSEN & MOLICH 1990). NIELSEN & MOLICH bewerteten dies wie folgt: „*Even finding some problems is of cause much better than finding no problems.*" (1990 S. 255).

### 2.4.5 Einige grundsätzliche Bemerkungen zu Reviews

**Weitere Fragestellungen**

Wir wollen zum Schluß dieses Unterkapitels einige grundsätzliche Fragestellungen aufwerfen, die sich aus dem Kapitel 2.4 ergeben, ohne sie abschließend zu behandeln. Dies erscheint uns sinnvoll, weil wir einige der Fragestellungen später noch einmal aufgreifen werden, wenn es um ein konkretes System zur Umsetzung von Ad-hoc Usability-Tests geht. Außerdem vermuten wir schon an dieser Stelle, daß ein verteiltes System für Online-Usability-Tests durch einfache Inspektions-Verfahren ideal ergänzt werden kann.

#### Welches spezifische Wissen über Software-Ergonomie sollen ExpertInnen und EntwicklerInnen haben?

**Rolle des Wissens beim Problemlösen**

Die Frage des Faktors Wissen in der Problemlösung war schon Mitte der achtziger Jahre ein Diskussionsgegenstand unter PädagogInnen und PsychologInnen. Polson und Jeffries betonten bereits seinerzeit sehr stark die Bedeutung einer wohlorganisierten, bereichsspezifischen Wissensbasis für das Problemlösen (vgl. MANDL, FRIEDRICH & HRON 1986).

**Wissen vs. Erfahrung**

Nielsen wies bei der Heuristischen Evaluation nach, daß fundiertes Expert-Innenwissen die Quote gefundener Fehler deutlich steigen läßt. Anders argumentieren KARAT J. & DAYTON (1995). Sie meinen, daß vor allem Erfahrung und Geschick vonnöten sind, um gute ergonomische Oberflächen zu erhalten: „*Unfortunately, the collection of things we know about human cognition does not form a very useful user-interface design guide.* (KARAT J. & DAYTON 1995 S. 164). Da die Benutzbarkeit von Software immer noch eine eher unbedeutende Rolle innerhalb des Software-Engineering spielt, plädieren die Autoren dafür, die EntwicklerInnen über Workshops so zu qualifizieren, daß sie selbst einfache Evaluationen vornehmen können, ohne über dezidiertes Fachwissen in Software-Ergonomie zu verfügen.

#### Individuelle vs. Gruppenbewertungen

**Teamwalkthroughs**

Teamwalkthroughs hatten in einigen Bereichen bessere Ergebnisse zu verzeichnen als individuelle Walkthroughs. Psychologen meinen, daß Gruppen selten bessere Leistungen erzielen als ihr bestes Mitglied (McGrath, J.E.: Groups: Interaction and performance Prentice Hall 1988 zitiert nach: KARAT C.-M., CAMPBELL & FIEGEL 1992 S. 397). Im Gegensatz dazu weist KARAT (KARAT C.-M., CAMPBELL & FIEGEL 1992, KARAT C.-M. 1994) nach, daß bespielsweise bei Walkthroughs Gruppen-*walkthroughs* bessere Ergebnisse erzielen als Einzel-Walkthroughs. Einige Evaluationsverfahren machen erst Sinn, wenn mehrere EvaluatorInnen unabhängig voneinander die Benut-

zungsoberfläche inspizieren. So werden für die Heuristische Evaluation in Abhängigkeit zum vorhandenen Fachwissen drei bis fünf ExpertInnen vorgeschlagen.

### Kosten von Reviews und Praxisnähe

**Hohe Kosten bei Labortests**

Die Diskussion um *usability inspection* ist unter anderem wegen der hohen Kosten von Labortests entstanden (vgl. MACK & NIELSEN 1994 S. 19). Verschiedene Studien zeigen, daß vor allem die Heuristische Evaluation extrem kostengünstig ist (vgl. JEFFRIES et al. 1991, KARAT C.-M. 1994).

**Produktorientiert**

Eine weitere Diskussion ist die Produktorientiertheit bzw. Praxisorientiertheit vor allem der Heuristischen Evaluation (vgl. MULLER et al. 1995 S. 115). Dabei wollen wir die Aspekte aus Kapitel 2.3.2 nicht beiseite schieben, aber eine vollständige, ganzheitliche Evaluation von Software werden sich auch in naher Zukunft nur die großen Betriebe leisten können.

**Discount Usability**

*Usability inspection methods* sind deshalb so attraktiv, weil sie teilweise schon von den EntwicklerInnen selbst „schnell" durchgeführt können (vor allem *cognitive walkthroughs* aber auch Heuristische Evaluation).

### Wieviele und welche Fehler werden tatsächlich korrigiert?

**Impact Ratio**

SAWYER, FLANDERS & WIXON (1996) untersuchten, wie effektiv *usability inspections* sind. Um dies zu überprüfen, benutzen sie eine Metrik, die sie *impact ratio* nannten. Bei dieser Metrik geht es darum, die gefundenen Fehler mit den Fehlern zu vergleichen, die tatsächlich nach einem Review behoben wurden.

$$impactRatio = \left( \frac{problemsComittedToFix}{totalproblemsFound} \right) * 100$$

**Durchschnitt des impact ratio**

In einer Studie untersuchten sie zehn Produkte, an denen Evaluationen mit Hilfe von *usability inspection*-Methoden durchgeführt wurden. Sie fanden heraus, daß der *impact ratio* im Durchschnitt 78 % war, allerdings bei einer Streuung von 58 % - 96 %. Sie untersuchten außerdem welche Fehler im wesentlichen behoben wurden, und fanden heraus, daß bei einer Skala von leichten, mittleren und schweren Fehlern das *impact ratio* 72 %, 72 % und 71 % betrug. Diese Studie zeigt vor allem, daß nicht die Schwere der Fehler ausschlaggebend für eine Korrektur ist, sondern Faktoren wie Zeitaufwand, Kostenfaktor oder als Risiko der Änderung.

**Warum werden Fehler nicht korrigiert?**

In der gleichen Studie nennen die Autoren Gründe, warum verschiedene Fehler nicht korrigiert werden (vgl. SAWYER, FLANDERS & WIXON 1996 S. 380):

- Das Entwicklungsteam hatte nicht genug Zeit oder Ressourcen, die Änderungen zu implementieren.
- Die Änderungen zu implementieren gefährdete die Performance des Produkts.
- Das Produkt wurde nicht weiter gepflegt.
- Das Problem existierte nur im Prototyp und nicht im tatsächlichen Produkt.

**Bestätigung durch andere Studien**

Auch andere Studien kamen zu ähnlichen Ergebnissen. Eine Studie der Hewlett-Packard-Company ergab bei 14 Produkten mit durchschnittlich 76 software-ergonomischen Mängeln (Streuung zwischen 25 und 149) eine Fehlerbehebungsrate mit einer *impact ratio* von 74 %[25] (vgl. GUNN 1995).

## 2.5 Usability-Test

**Definitionen für Usability-Test**

Im Gegensatz zu den in Kapitel 2.4 vorgestellten Review-Verfahren basieren die Usability-Tests nicht auf ExpertInnenwissen, sondern auf der Erfahrung und dem Wissen der BenutzerInnen: „*The focus of a usability test is the user s experience with a product or process.*" (USABILITY MANAGEMENT 1998 S. 2). Der Umfang oder die Art der Durchführung dieser Tests ist dabei unbestimmt. Es werden keine Methoden vorgeschrieben: „*The term test is a broad term that encompasses any method for assessing whether the goals have been met.*" (WIXON & WILSON 1997 S. 669).

Die Ziele des Usability-Tests müssen während des Tests aufgestellt und ihr Erreichen durch unterschiedliche Daten beurteilt werden: „*During this procedure the tester collects both quantitative and qualitative data regarding the user s performance and satisfaction.*" (BRANAGHAN S. 1).

**Ziele des Usability-Tests**

Der zentrale Gegenstand der Usability-Tests, die BenutzerInnenleistung und -zufriedenheit basiert auf der in Kapitel 2.2 vorgestellten Erläuterung für Usability. Die Erfüllung dieser Anforderungen wird durch folgende Definition konkretisiert: „Usability-Tests konzentrieren sich auf die Ermittlung, ob ein Produkt leicht erlernbar, zufriedenstellend bei der Benutzung ist und die gesamte Funktionalität enthält, die die BenutzerInnen wünschen."[26] (BRANAGHAN S. 1).

**Stellenwert des Usability-Tests**

Diese Art der Evaluation von Software besitzt einen hohen Stellenwert innerhalb des Evaluationbereichs:

„*User testing with real user is the most fundamental usability method and is in some sense irreplaceable, since it provides direct information about how people use computers and what their exact problems are with the concrete interface being tested.*" (NIELSEN 1993 S. 165).

### 2.5.1 Durchführung eines Usability-Tests

**Usability-Test Durchführung**

Unabhängig von konkreten Methoden, die während eines Usability-Tests eingesetzt werden sollen, hat sich in der Praxis ein Verfahren durchgesetzt, das den Ablauf eines Tests beschreibt:

**Usability-Test als mehrstufiger Prozeß**

„Usability-Test ist ein mehrstufiger Prozeß, der Testvorbereitungen, den tatsächlichen Test, die Analyse der Testresultate, die Modifikation des Software-Systems und das wiederholte Testen des Systems umfaßt."[27] (NASA). In der Literatur werden verschiedene Ausprägungen dieses Prozesses beschrieben. Die folgende Auflistung, die eine Basis für alle Beschreibungen darstellt (vgl. BRANAGHAN 1997 S. 2ff, SPOOL, SNYDER & ROBINSON 1996, BRANAGHAN 1997, NIELSEN 1993 S. 170ff, NIELSEN 1994b S. 5, WIXON & WILSON 1997

[25] Streuung zwischen 50 % bis 95 %.

[26] Eigenübersetzung.

[27] Eigenübersetzung.

S. 669ff, NASA, HANSEN 1991), nennt die einzelnen Stufen dieses Prozesses, dessen Aufgaben, Ziele und Resultate:

1. **Spezifikation des Zwecks/Ziels des Tests**

**Zweck des Usability-Test**

Am Anfang der Vorbereitung eines Usability-Tests muß die Frage beantwortet werden, zu welchem Zweck der Test durchgeführt werden soll. Analog zu den aufgestellten Zielen einer Evaluation (vgl. Kapitel 2.3.1) wird in diesem Schritt die Richtung des Usability-Tests festgelegt (vgl. BRANAGHAN S. 2f, HOLZ AUF DER HEIDE 1993 S. 160f):

- Vergleichender Test
  Wird das zu betrachtende Produkt mit einem Konkurrenzprodukt in bezug auf Usability verglichen?
- Analysierender Test
  Sind beispielsweise die Hilfestellungsanfragen für ein bestimmtes, eventuell neu eingeführtes Produkt enorm gestiegen, so daß eine grundsätzliche Usability-Analyse diese Produktes durchgeführt werden muß?
- Bewertender Test
  Wurde ein neues Produkt entworfen, und soll dieser Entwurf vor seiner weiteren Einführung auf Usability bewertet werden (vgl. NASA, SULLIVAN 1996 S. 2)?

Die Bestimmung des Zwecks eines Usability-Tests hat entscheidende Auswirkungen auf das weitere Vorgehen, insbesondere auf die spätere Auswahl der konkreten Methoden (vgl. NIELSEN 1993 S. 170).

**Ziele des Tests als Fragenkatalog**

Die Ziele des Tests werden in Form eines Fragenkataloges aufgestellt, der mittels des Tests beantwortet werden soll. Die Fragen müssen möglichst präzise und eindeutig formuliert sein (vgl. BRANAGHAN S. 3).

2. **Bestimmung des BenutzerInnenprofils**

**Eigenschaften und Inhalte eines BenutzerInnenprofils**

Die Personen, die während des Tests die Rolle der BenutzerInnen übernehmen, müssen die tatsächlichen, späteren BenutzerInnen des Systems möglichst repräsentieren:

„*The main rule regarding test users is that they should be as representative as possible of the intended users of the system.*“ (NIELSEN 1993 S. 175).

Das Ziel des Tests kann durch die falsche Auswahl der Personen verfälscht bzw. verfehlt werden:

„*When you test the wrong participants, you are in effect testing the wrong human-machine-system.*“ (BRANAGHAN 1997 S. 1).

Die richtigen TeilnehmerInnen werden gefunden, wenn in diesem Schritt ein BenutzerInprofil entsprechend dem Produkt erstellt wird. Dieses Profil besteht aus einer Beschreibung der Charakteristiken der BenutzerInnen, die das Produkt (später) benutzen werden. Da es nicht die „durchschnittliche“ BenutzerIn gibt, muß auf eine prozentuale Verteilung der Teilnehmenden gemäß den Charakteristiken geachtet werden (vgl. BRANAGHAN S. 3). Die folgende Liste gibt Beispiele für

diese Charakteristiken (vgl. NIELSEN 1993 S. 43f und S. 175f, BRANAGHAN 1997 S. 1f, BRANAGHAN S. 3, WIXON & WILSON 1997 S. 663f, NASA):

- Computererfahrung
- Bildungsniveau
- Alter
- Geschlecht
- Ausdrucksfähigkeit (Muttersprache)

**Verbreitung des Einsatzes eines BenutzerInnenprofils**

Die Anwendung eines BenutzerInnenprofils ist im Rahmen des Usability-Tests noch nicht sehr verbreitet. Laut einer Umfrage (n=25) von WIXON & WILSON (1997 S. 663) arbeiten nur 60 % der befragten Projekte überhaupt (28 % fast immer, 8 % sehr regelmäßig, 8 % regelmäßig und 16 % manchmal) mit explizitem BenutzerInnenprofil, während 40 % dieses Konzept selten (24 %) oder gar nicht (16 %) anwenden.

3. **Ermittlung der Anzahl der BenutzerInnen**

**Anzahl der BenutzerInnen**

Neben den Charakteristiken der BenutzerInnen spielt die Anzahl der TestteilnehmerInnen eine wichtige Rolle. Laut NIELSEN & LANDAUER (1993 S. 210) gibt es einen direkten Zusammenhang zwischen der Anzahl der BenutzerInnen und der gefundenen Usability-Probleme.

**Verhältnis zwischen Anzahl der BenutzerInnen und Usability-Problemen**

Nach verschiedenen Studien haben NIELSEN & LANDAUER (1993) folgende Formel für das Verhältnis von gefundenen Usability-Problemen und der Anzahl der TestteilnehmerInnen aufgestellt (vgl. Kapitel 2.4.4, BRANAGHAN 1997 S. 3, BRANAGHAN S. 3f):

$$Found(i) = N(1-(1-l)^i)$$

4. **Identifikation der Aufgaben**

**Phasen der Aufgabenidentifizierung**

Die Erkennung und Erstellung von konkreten Arbeitsaufgaben besteht aus vier Phasen (vgl. BRANAGHAN S. 5):

Phase 1: Beschreibung der Aufgabe
Die Aufgabe muß kurz beschrieben werden, da sie den späteren Einsatz des Systems repräsentieren muß (vgl. NIELSEN 1993 S. 185).

Phase 2: Materialbeschreibung
Nachdem die Aufgabe beschrieben ist, müssen die Voraussetzungen (z. B. bestimmte Konfigurationen) und benötigten Materialien (z. B. Kommunikationsverbindungen) für den Test aufgelistet werden.

Phase 3: Erfolgsbeschreibung
Jede Arbeitsaufgabe endet mit dem Erreichen eines bestimmten Zustandes, der in dieser Phase beschrieben werden muß. Anhand dieser Beschreibung kann der Erfolg oder Mißerfolg der Arbeitsaufgabe bestimmt werden.

Phase 4: Zeitbestimmungen
Jede Arbeitsaufgabe muß innerhalb eines definierten Zeitrahmens bearbeitet und beendet werden.

Folgende Liste beschreibt mögliche Aufgaben für einen Usability-Test eines fiktiven Text-Editors (vgl. BRANAGHAN S. 5):

**Beispiel einer Aufgabenbeschreibung**

| Aufgabenbeschreibung | Vorbedingungen und Material | Erfolgsindikatoren | Zeitbeschreibungen |
|---|---|---|---|
| Öffne das Dokument mit dem Titel „Prozeduren“ und melde die erfolgreiche Bearbeitung. | Auf dem Rechner sollte die aktuelle Version des Texteditors laufen. | Die BenutzerIn öffnet das Dokument erfolgreich und meldet dies. | 3 Minuten |
| Drucke das Dokument „Prozeduren“ mit der besten Druckqualität und melde die erfolgreiche Bearbeitung. | Auf dem Rechner sollte die aktuelle Version des Texteditors laufen und das Dokument sollte geöffnet sein. | Der Drucker druckt das Dokument in bester Qualität. | 4 Minuten |

Tabelle 1: Aufgabenbeschreibung

5. **Bestimmung der Leistungs- und Zufriedenheitsmetriken**

**Qualitative und quantitative Daten**

In Abhängigkeit von den Arbeitsaufgaben müssen die Metriken der gesammelten Daten klassifiziert und quantifiziert werden. Da der Schwerpunkt auf BenutzerInnenleistung und -zufriedenheit liegt, wird folgende Einteilung vorgenommen (vgl. NASA, BRANAGHAN S. 5):

- Leistung (quantitative Daten):
  Die Zeit, die benötigt wurde, um die Aufgabe durchzuführen, ist ein Beispiel für qualitative Daten.
- Zufriedenheit (qualitative Daten):
  Die Benutzerzufriedenheit bei der Durchführung der Aufgabe ist ein Beispiel für qualitative Daten.

Diese Metriken dienen als Grundlage für die spätere Auswahl und den Einsatz der konkreten Testmethoden.

6. **Spezifikation der Rolle der Testmitglieder**

**Rolle der Testteilnehmenden**

In diesem Schritt muß die Rolle aller Teilnehmenden am Test konzipiert werden. Diese Gruppe läßt sich in vier Kategorien einteilen (vgl. BRANAGHAN S. 5, NASA):

- BenutzerInnen
  Diese Personen repräsentieren gemäß den aufgestellten BenutzerInnenprofilen die späteren BenutzerInnen des Systems, führen die Aufgaben durch und bewerten das System an sich.[28]
- TestbegleiterInnen[29]
  Diese Personen stellen die direkte Verbindung zwischen den BenutzerInnen und dem Testteam dar. Vor Testbeginn begrüßt diese Person die BenutzerInnen, begleitet sie während des Tests, ohne

[28] Ethische Aspekte des Einsatzes von Menschen in Usability-Tests finden sich bei NIELSEN (1993 S. 181ff).

[29] *test monitor*.

sie bei der Bearbeitung zu beeinflussen, und verabschiedet sie nach Beendigung der Aufgaben.

- TestaufzeichnerInnen
  Diese Personen sammeln alle Daten (Zeiten, Fehler, Kommentare, etc.), die für die einzelnen Aufgaben vorgesehen sind.

- TestbeobachterInnen
  Diese Personen sind für das eigentliche Produkt verantwortlich (z. B. EntwicklerInnen) und daran interessiert zu erfahren, wie die eigentlichen BenutzerInnen mit ihrem Produkt umgehen; es besteht die Möglichkeit, mittels des Testbegleitenden Fragen an die BenutzerInnen zu stellen.

  Laut NIELSEN sollten die Personen[30], die den Test leiten, folgende Qualifikationen besitzen (1993 S. 179f):

  - Erfahrungen mit den eingesetzten Methoden
  - Umfassende Kenntnisse des Produktes und seiner Oberfläche

  Bei der Auswahl der TestbegleiterInnen, TestaufzeichnerInnen und TestbeobachterInnen wird in der Literatur (vgl. NIELSEN 1993 S180f, BRANAGHAN S. 6, NASA) darauf hingewiesen, daß die eigentlichen EntwicklerInnen nicht Bestandteil des Testteams sein sollten (außer als TestbeobachterInnen), da mehrere Gründe diesem Einsatz entgegenstehen:

  - Sie sind zu sehr mit dem Design des Produktes beschäftigt, als das sie objektiv urteilen könnten.
  - Es besteht die latente Gefahr, daß sie ihr Produkt nur demonstrieren wollen, statt die BenutzerInnen das Produkt benutzen zu lassen.
  - Sie sind versucht, direkt einzugreifen, wenn die BenutzerInnen Probleme bei der Bearbeitung ihrer Aufgaben haben.

  Darüber hinaus sollten die EntwicklerInnen aber „bei der Planung und Überwachung der Probleme, die der Test aufdeckt, beteiligt sein." (BRANAGHAN S. 6)[31].

7. **Bestimmung des Test-Einrichtungslayouts**

**Aufbau und Einrichtung einer Usability-Umgebung**

„Viele Usability-Tests werden in speziell ausgestatteten Usability-Laboren durchführt. ... spezielle Labore sind zwar bequem, aber nicht absolut notwendig für Usability-Tests."[32] (NIELSEN 1993 S. 200)[33]. Folgender Aufbau eines Labors wird als typisch beschrieben (vgl. NIELSEN 1993 S. 201, NASA; BRANAGHAN S. 6, ROWLEY 1994 S. 254, NEUGEBAUER & SPIELMANN 1993 S. 328):

[30] IELSEN bezeichnet sie als *experimenters*.
[31] Eigenübersetzung.
[32] Eigenübersetzung.
[33] Eigenübersetzung.

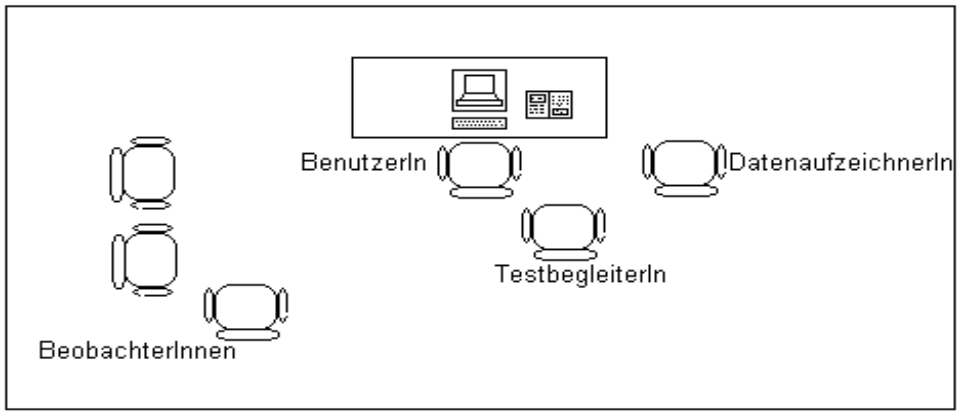



Abbildung 5: Typisches Test-Einrichtungslayout

**Standard-Ausstattung der Usability-Labore**

NIELSEN ermittelte in einer Studie (n=13) die typische Ausstattung eines Usability-Labors (1993 S. 202f, 1994b):

- 63 $m^2$ Laborfläche mit 12 $m^2$ Testraumfläche
- Einwegspiegel (92 % aller Labore)
- Schallschutz
- Verschiedene Räume für Beobachter, Datensammler und Benutzer (im Durchschnitt 5.8)
- Mehrere Videokameras (im Durchschnitt 2.2)
- Scan-Konverter (46 % aller Labore)

8. **Durchführung eines Pilot-Tests**

**Faktoren für Testdurchläufe**

„Kein Usability-Test sollte durchgeführt werden, ohne den Testablauf an verschiedenen Testdurchläufen zu prüfen.“[34] (NIELSEN 1993 S. 174). Folgende Faktoren des Tests können so überprüft werden (vgl. NIELSEN 1993 S. 174f, BRANAGHAN S. 7, WIXON & WILSON 1997 S. 672):

- Funktionieren des Testequipments
- Verständlichkeit der Aufgabenbeschreibungen
- Verhältnis von Aufgaben und Zeitbeschreibungen
- Aussagekraft der aufgestellten Metriken

9. **Durchführung des Tests**

**Durchführung des Tests**

Der folgende Überblick beschreibt den Ablauf eines Usability-Tests (vgl. BRANAGHAN S. 7, WIXON & WILSON 1997 S. 672f, NIELSEN 1993 S. 187ff):

a) Begrüßung der BenutzerInnen und Informationen über die Umgebung (Toiletten, Ausgänge, etc.)

b) Veranschaulichung der Testumgebung (Laborumgebung)

[34] Eigenübersetzung.

c) Erläuterung der eingesetzten Methoden (z. B. Lautes Denken, Videoaufzeichnungen)

d) Kurze Lagebesprechung als Einführung zur eigentlichen Aufgabe

e) Rückfragen der BenutzerInnen ermitteln und beantworten

f) Beginn der eigentlichen Aufgabe

g) Nachfragen nach jeder Aufgabe zu speziellen Probleme bei der Bearbeitung

h) Nach Beendigung aller Aufgaben Ermittlung der BenutzerInnenkommentare

i) Danksagung und Überreichung von Dankpräsenten

10. **Analyse der Daten**

**Ergebnisse**

In der letzten Phase des Tests werden die ermittelten Daten analysiert und ausgewertet. Die konkreten Probleme der BenutzerInnen werden gemäß der aufgestellten Metriken klassifiziert (vgl. WIXON & WILSON 1997 S. 674f):

a) Sind die quantitativen Ziele des Tests erreicht worden?

b) Welche Usability-Probleme besitzt das Produkt?

c) Wie schwerwiegend sind diese Probleme?

Anschließend können die ermittelten Daten entsprechend ihrer Schwere und dem Aufwand ihrer Beseitigung geordnet werden (vgl. NASA). Die Ergebnisse werden anschließend dem EntwicklerInnenteam präsentiert und dort diskutiert (vgl. WIXON & WILSON 1997 S. 675).

### 2.5.2 Usability-Testmethoden

Es existieren grundsätzlich verschiedene Methoden, die Ziele des Usability-Tests zu überprüfen (vgl. NIELSEN, NIELSEN 1993, KARAT J. 1997, SPOOL, SNYDER & ROBINSON 1996):

1. **Lautes Denken**

**Wichtigste Usability-Engineering-Methode**

Lautes Denken stellt die wertvollste Usability-Engineering-Methode dar. Durch in Worte gefaßte Gedanken ermöglicht die BenutzerIn zu verstehen, wie das System gesehen wird. Man erhält ein direktes Verständnis, welche Teile des Dialogs Probleme aufwerfen (vgl. NIELSEN 1993 S. 195).

**Ursprung in der Psychologie**

Lautes Denken kommt traditionell aus der Psychologie und wurde in die praktische Evaluation von Mensch-Computer-Schnittstellen übertragen.

**Äußerungen während des Gebrauchs**

Während des Tests muß darauf geachtet werden, daß die BenutzerInnen ihre Äußerungen machen, während sie das System benutzen. Dies schließt eine anschließende Rationalisierung aus, die die gewünschten Daten verfälschen würde (vgl. NIELSEN 1993 S. 196, KARAT J. 1997 S. 695).

**Anzahl und Fülle der Daten**

Der Vorteil des Lauten Denkens ist die Anzahl und Fülle der qualitativen Daten, die schon von einer geringen Anzahl von BenutzerInnen ermittelt werden können (vgl. NIELSEN 1993 S. 195).

**Keine direkte Leistungsmessung**

Hauptnachteil dieser Methode ist, daß sie sich nicht für Leistungsmessungen eignet (vgl. NIELSEN 1993 S. 195). Allerdings sollte sie im Zusammenhang mit anderen Methoden eingesetzt werden, die eine Leistungsmessung als Gegenstand haben (vgl. KARAT J. 1997 S. 695).

**Gefahr der Verfälschung**

Beim Lauten Denken besteht die Gefahr der Verfälschung der gesammelten Daten, wenn die BenutzerInnen ihre eigenen Theorien über die Gründe des Problems auf die Arbeit mit dem System übertragen. Daten dieser Art dürfen in der Auswertung nicht zu stark berücksichtigt werden (vgl. NIELSEN 1993 S. 195).

**Keine Lenkung durch die TestbegleiterInnen**

Weiterhin dürfen die TestbegleiterInnen nicht lenkend in den Test eingreifen (vgl. NIELSEN 1993 S. 197, KARAT J. 1997 S. 695f), damit die Ergebnisse nicht verfälscht werden. Aus diesen Gründen sollten die TestbegleiterInnen Erfahrungen mit der Methode und mit experimenteller Psychologie besitzen (vgl. KARAT J. 1997 S. 695).

**Unnatürlichkeit des Lauten Denkens und Notwendigkeit für Probedurchläufe**

Aus dieser Erfahrung heraus könnten die TestbegleiterInnen diese Methode den BenutzerInnen näherbringen, da Lautes Denken für viele Menschen unnatürlich und deshalb schwer durchzuführen ist. Dies gilt insbesondere für erfahrene BenutzerInnen, die Arbeitsabläufe schon unbewußt automatisiert haben (vgl. NIELSEN 1993 S. 196). Durch verschiedene Probetests mit anderen Inhalten als der eigentlichen Aufgabe kann den BenutzerInnen das Prinzip des Lauten Denkens verdeutlicht werden (vgl. NIELSEN 1993 S. 197).

**Veränderung des Lernverhaltens**

Die Anwendung des Lauten Denkens verlangsamt das eigentliche Arbeiten, da man sich auch darauf konzentrieren muß, was man verbalisiert (vgl. NIELSEN 1993 S. 196). Lautes Denken führt aber gleichzeitig dazu, daß man sich Gedanken über das eigene Modell des Systems macht und dabei Unstimmigkeiten und Inkonsistenzen mit seinen Vorstellungen feststellt. In verschiedenen Untersuchungen wurde ermittelt, daß genau dieses Verhalten dazu führt, daß neue Oberflächen bis zu 9 % schneller und mit 20 % weniger Fehlern erlernt werden (vgl. NIELSEN 1993 S. 196).

### 2. Konstruktive Interaktion[35]

**Variation des Lauten Denkens**

„Eine Variation des Lauten Denkens wird Konstruktive Interaktion genannt, bei der zwei BenutzerInnen zusammen das System benutzen.“ [36] (NIELSEN 1993 S. 198f). Sie können miteinander sprechen und sich austauschen.

**Natürlicher als Lautes Denken**

Der Vorteil dieser Methode gegenüber dem Lauten Denken ist, daß sie natürlicher ist, da Menschen es gewohnt sind, mit jemanden zu sprechen, wenn sie Probleme haben und eine Lösung suchen (vgl.

[35] Auch *codiscovery learning.*

[36] Eigenübersetzung.

NIELSEN 1993 S. 198). Aus diesem Grunde machen sie eher Kommentare bezüglich des Systems als beim Lauten Denken.

**Probleme bei unterschiedlichen BenutzerInnen**

Ein Nachteil dieser Interaktion zeigt sich, wenn die beteiligten BenutzerInnen verschiedene Strategien bei dem Umgang mit Computern haben. Dies kann dazu führen, daß der Test zu keinem verwertbaren Ergebnis führt, da die gemachten Aussagen zu stark variieren (vgl. NIELSEN 1993 S. 198).

3. **Betreuer-Methode**

**Weitere Variation des Lauten Denkens**

Eine weitere Variation der Lautes Denken-Methode ist die Betreuer-Methode[37], bei der eine direkte Kommunikation zwischen BenutzerInnen und TestbegleiterInnen erwünscht ist. Dies steht im Gegensatz zu den anderen Methoden, bei denen die TestbegleiterInnen möglichst wenig mit den BenutzerInnen interagieren sollen (vgl. NIELSEN 1993 S. 199f). Während des Tests dürfen die BenutzerInnen alle systembetreffenden Fragen stellen, die die TestbegleiterInnen nach bestem Wissen beantworten müssen.

**Unterstützung von unerfahrenen BenutzerInnen**

Ein Vorteil dieser Methode ist die Unterstützung von systemunerfahrenen BenutzerInnen, um einerseits Kenntnisse über das System schneller zu vermitteln und andererseits Hinweise auf die Schulung und Dokumentation zu bekommen, die notwendig sind, um mit dem System arbeiten zu können (vgl. NIELSEN 1993 S. 200).

4. **Daten-Aufzeichnungen (Data logging)**

**Automatische Erstellung von Statistiken**

„*Logging involves having the computer automatically collect statistics about the detailed use of the system. Logging the user s actual use of the system is particulary useful because it shows how users perform their actual work.*“ (NIELSEN 1993 S. 217).

**Funktionalitätsbewertung**

Mittels dieser Statistiken kann überprüft werden, welche Funktionalitäten oft und welche seltener benutzt werden. Dies kann dazu führen, daß Funktionen leichter erreichbar oder komplett aus dem System entfernt werden.

**Rekonstruktion des Systemverhaltens**

Außerdem besteht die Möglichkeit, das Systemverhalten zu einem späteren Zeitpunkt zu überprüfen, zu dem komplette Aufzeichnungen abgespielt werden. Mit dieser Rekonstruktion kann das Systemverhalten in bezug auf die Aktivitäten der BenutzerInnen untersucht werden (vgl. NIELSEN 1993 S. 219).

**Verletzung der Privatssphäre**

Da die automatische Protokollierung des Systemverhaltens eine Verletzung der Privatsphäre der BenutzerInnen darstellt, müssen sie darauf hingewiesen werden, welche Statistiken zu welchem Zweck erzeugt werden. Weiter muß ihnen versichert werden, daß diese Ergebnisse anonym bleiben (vgl. NIELSEN 1993 S. 219). Dies gilt natürlich auch für alle anderen Tests.

[37] c*oaching method.*

Weitere Aspekte und Einsatzmöglichkeiten der Aufzeichnungsmethode finden sich in der Literatur (vgl. NIELSEN 1993 S. 217ff, DIX et al. 1995 S. 453f, HOLZ AUF der HEIDE 1993 S. 165ff).

5. **Interviews/Fragebogen**

**Benutzermeinung erfragen**

„*Many aspects of usability can best be studied by simply asking the users. From a usability perspective, questionnaires and interviews are indirect methods, since they do not study the user interface itself but only user s opinion about the user interface.*" (NIELSEN 1993 S. 209).

**Äquivalenz von Interviews und Fragebogen**

Während der Interviews bzw. in den Fragebogen beantworten BenutzerInnen Fragen, und zeichnen die Antworten auf. Prinzipiell sind Interviews und Fragebogen gleichwertig. Folgende Unterschiede sind jedoch erkennbar (vgl. NIELSEN 1993 S. 210ff):

- Fragebogen müssen bei der Bearbeitung nicht begleitet werden, während Interviews von einem Interviewer durchgeführt werden müssen.
- Interviews sind im Gegensatz zu starren Fragebogen flexibler, da die InterviewerInnen die Möglichkeit haben, bei komplexen Fragen die Inhalte näher und detaillierter zu erläutern. Daher sollten Fragebogen vor ihrem Einsatz in Pilot-Durchläufen daraufhin getestet werden, ob sie die erwarteten Daten überhaupt liefern können.

Fragebogen können an alle beteiligten BenutzerInnen vergeben werden, um beispielsweise spezifische Unterschiede zwischen BenutzerInnengruppen erkennen zu können.

Detaillierte Informationen zu Fragebogen finden sich in der Literatur (vgl. NIELSEN 1993 S. 209ff, KARAT J. 1997 S. 696f, DIX et al. 1995 S. 457ff).

6. **Beobachtung/Video-Aufzeichnung**

**Einfache Methode**

„*Simply visiting the user to observe them work is an extremely important usability method with applications both for task analysis and for information about the true field usability of installed systems.*" (NIELSEN 1993 S. 207ff). Diese Methode ist eine der einfachsten in bezug auf Vorbereitung und Durchführung, da die BenutzerInnen einfach nur beobachtet werden. Während der Beobachtung können Notizen gemacht bzw. die gesamte Beobachtung auf Video aufgezeichnet werden.

**Hoher Aufwand bei der Auswertung**

Einen wesentlichen Nachteil der Videoaufzeichnung stellt die anschließende Analyse und Auswertung der aufgezeichneten Arbeiten dar: Die Auswertung der Aufzeichnungen benötigt drei- bis zehnmal mehr Zeit als die Dauer der eigentlichen Aufzeichnung. (vgl. NIELSEN 1993 S. 203, SPOOL, SNYDER & ROBINSON 1996 S. 366, DIX et al. 1995 S. 453).

**Notwendigkeit von Videoaufzeichnungen**

Ob eine Videoaufzeichnung bzw. deren anschließende Auswertung überhaupt notwendig ist, hängt von Inhalt und Ziel des Usability-Tests ab: „*For practical usability engineering purposes, however, there is normally no need to review a user test on videotape since one is mostly interested in finding the*

*major usability catastrophes anyway. These usability problems tend to be so glaring that they are obvious the first time they are observed and therefore do not require repeated persual of a record of the test session.*" (NIELSEN 1993 S. 203).

Ist das Ziel jedoch, eine formale Auswirkungsanalyse[38] der Usability-Probleme durchzuführen, so benötigt man komplette Aufzeichnungen. Der Ablauf dieser Analysen erfordert es, in einem ersten Schritt die eigentlichen Usability-Probleme zu erkennen, um anschließend in einem zweiten festzustellen, wie viele BenutzerInnen durch diese Probleme bei ihrer Arbeit behindert wurden (vgl. NIELSEN 1993 S. 204).

Videoaufzeichnungen können auch eingesetzt werden, um EntwicklerInnen und ManagerInnen davon zu überzeugen, daß Usability-Probleme wirklich einen Mangel darstellen: „Videoaufzeichnungen dienen als wesentliches Kommunikations-Medium in vielen Organisationen[...]. Ein Video sehen, in dem eine BenutzerIn mit einem Problem kämpft, überzeugt diese Menschen."[39] (NIELSEN 1993 S. 204).

Weitere Hinweise und Möglichkeiten der Videoaufzeichnungen finden sich in der Literatur (vgl. NIELSEN 1993 S. 203ff, DIX et al. 1993 S. 453, HOLZ AUF DER HEIDE 1993 S. 165ff).

Für alle vorgestellten Methoden hat Nielsen folgenden Hinweis gegeben, der bei der Auswertung der Daten beachtet werden muß (vgl. NIELSEN 1993 S. 209f):

„Man kann Benutzeräußerungen nicht unbesehen glauben. Daten über das tatsächliche Verhalten der BenutzerInnen sollten Vorrang haben vor den Äußerungen der Gedanken über das, was sie tun."[40] (NIELSEN 1993 S. 29) ... "die Benutzer dachten, daß sie die Anweisungen verstanden hatten, obwohl sie sie tatsächlich nicht verstanden hatten."[41] (NIELSEN 1993 S. 210).

**Vor- und Nachteile der vorgestellten Methoden**

Die folgende Zusammenfassung gibt einen Überblick über die Methoden, die die Anzahl der benötigten BenutzerInnen und die herausragenden Vor- und Nachteile der Methoden (vgl. NIELSEN 1993 S. 224):

| Methoden-bezeichnung | Benutzer-Innen-Anzahl | Vorteile | Nachteile |
|---|---|---|---|
| Lautes Denken | ~3 - 5 | • Zeigt Mißverständnisse der BenutzerInnen<br>• Billiger Test | • Unnatürlich für die meisten Benutzer-Innen |
| Data-Logging | >20 | • Zeigt wenig/häufig benutzte Funktionen<br>• Kann kontinuierlich betrieben werden | • Großes Datenaufkommen<br>• Verletzung der Privatssphäre |
| Interviews | ~5 | • Flexibel | • Zeitaufwendig<br>• Aufwendige Analyse und unsichere Vergleichsmöglichkeiten |

[38] f*ormal impact analysis.*
[39] Eigenübersetzung.
[40] Eigenübersetzung.
[41] Eigenübersetzung.

| Methoden-bezeichnung | Benutzer-Innen-Anzahl | Vorteile | Nachteile |
|---|---|---|---|
| Fragebogen | >30 | • Subjektive Benutzer-wünsche<br>• Leicht wiederholbar | • Pilot-Tests notwendig |

Tabelle 2: Methodenübersicht

Weitere Beschreibungen zu den Einsatzmöglichkeiten der genannten Methoden finden sich in der Literatur (vgl. KARAT J. 1997 S. 694, HOLZ AUF DER HEIDE 1993 S. 162).

Anhand der beschriebenen Vor- und Nachteile läßt sich erkennen, daß eine Kombination der vorgestellten Methoden die Vorteile summiert und einen Teil der Nachteile eliminiert: „*It is therefore highly recommended not to rely on a single usability method to the exclusion of others.*“ (NIELSEN 1993 S. 223).

## 2.6 Usability-Tests - Reviews: Ein Vergleich

**Evaluation muß sich Entwicklungsprinzipien anpassen**

Software-Entwicklungsteams arbeiten mit restriktiven Budgets und Zeitplänen sowie mit geringen personellen und technischen Ressourcen. Die Evaluation der Software muß sich diesen Zwängen beugen, d. h., sie soll kostengünstig sein, wenig Zeit beanspruchen, wenig Personal und wenig technische Ressourcen benötigen (vgl. KARAT C.-M. 1994 S. 203).

**Im Mittelpunkt stehen konkrete Ziele**

Verschiedene Evaluationsmethoden haben verschiedene Zielsetzungen: im Mittelpunkt stehen jedoch immer sehr konkrete Ziele, die mit bestimmten Arbeitsaufgaben verbunden sind (*Usability evaluation objectives*). Ziele in diesem Kontext sind beispielsweise (vgl. KARAT C.-M. 1994 S. 205):

- 90 % der BenutzerInnen sollen eine repräsentative Arbeitsaufgabe fehlerfrei nach dem dritten Versuch abschließen können.
- 80 % der BenutzerInnen sollen eine komplette Arbeitsaufgabe lösen, ohne Hilfestellung in Anspruch zu nehmen.
- BenutzerInnen sollen in der Lage sein, eine bestimmte Arbeitsaufgabe innerhalb 10 Minuten abzuschließen.

**Inspection Methods sehen nicht die gesamte Bedienungsoberfläche, sondern nur Teile**

Analysen zeigen, daß *inspection methods* im Gegensatz zu Usability-Tests nicht in der Lage sind, diese klaren Ziele zu evaluieren. *Usability inspections* decken verschiedene Probleme auf, wie Genauigkeits-, Systematik- oder Konsistenzprobleme. Wenn es darum geht, die Ganzheitlichkeit der Arbeitsaufgabe zu sehen, sind empirische Testmethoden im Vorteil (vgl. HAMPE-NETELER & RÖDIGER 1992, DESURVIRE 1994). Zu dem gleichen Schluß kommt auch BROOKS. Sie resümiert: „*Inspection methods are effective evaluation tools when they are used to meet objectives such as minimizing problems before user testing is undertaken.*“ (1994 S. 271).

### 2.6.1 Anzahl der Probleme

**Heuristische Evaluation findet die meisten Probleme**

Eine Studie von JEFFRIES et al. untersuchte vier Evaluationsmethoden (1991):

- Heuristische Evaluation
- *Cognitive walkthrough*
- *Guidelines walkthrough*
- Usability-Tests.

Das Ergebnis der Studie zeigt, daß Heuristische Evaluation im Vergleich dieser vier Methoden die meisten Probleme findet.

Von 206 unabhängigen Problemen fand Heuristische Evaluation mehr als die Hälfte (105). Usability-Tests entdeckten die zweithöchste Fehleranzahl. KARAT bemerkte, daß im Vergleich von Labortests mit *cognitive walkthrough* die Labortests deutlich effizienter sind, auch wenn *cognitive walkthroughs* im Team durchgeführt werden (vgl. KARAT C.-M., CAMPBELL & FIEGEL 1992).

| | **Heur. Eval.** | **Usability** | **Guidelines** | **Cog. Walk.** | **Total.** |
|---|---|---|---|---|---|
| **Underlying system** | 15 | 38 | 3 | 0 | 21 |
| **Evaluator error** | 7 | 0 | 0 | 3 | 10 |
| **Non-repeatable** | 6 | 3 | 0 | 2 | 11 |
| **Other** | 3 | 0 | 0 | 0 | 3 |
| **Core** | 121 | 32 | 35 | 35 | 223 |
| **Core no dupl.** | 105 | 31 | 35 | 35 | 206 |
| **total** | **152** | **38** | **38** | **40** | **268** |

Tabelle 3: Gefundene Fehler, unterschieden in Problemtyp und Evaluationsmethode (JEFFRIES et al. 1991 S. 121)

**Wie werden Probleme gefunden?**

In der Studie von JEFFRIES et al. (1991) wurde ebenfalls ausgewertet, wie die verschiedenen Probleme gefunden wurden:

- Durch die Methode selbst
- Durch Seiteneffekte[42]
- Bereits vor der Evaluation beim Kennenlernen des Systems.

Während bei Heuristischen Evaluationen alle Probleme durch Anwendung der Methode gefunden wurden, wurden beispielsweise bei *guidelines walkthroughs* bereits ein Drittel der Probleme vor dem eigentlichen Test, also zufällig, entdeckt (vgl. JEFFRIES et al. 1991 S. 121).

[42] Als Beispiel: Während eine Richtlinie des Layouts getestet wurde, wurde ein Problem in der Menüorganisation erkannt.

### 2.6.2 Schwere der Probleme

**Kategorisierung von Fehlern**

Jede Studie führt eine eigene Kategorisierung der Fehler in verschiedene Schweregrade (*problem severing rates)* durch. Meistens wurden die Fehler in drei Schweregrade unterteilt, teilweise auch in mehr. Bei den verschiedenen Studien wurde nicht genau deutlich, welche Fehlerklassen zu welchen Schweregraden zusammengefaßt werden. Fehlerklassen könnten sein: Kosmetische Fehler (falsch geschriebene Bezeichner), Inkonsistenzfehler (Unterschiedliche Bezeichnung von Schaltern: Abbruch vs. Abbrechen) bis hin zu Fehlern, bei denen die Arbeitsaufgabe nicht zu Ende geführt werden kann.

**Aufteilung der Fehler in Schweregrade**

JEFFRIES et al. (1991) teilten die verschiedenen Probleme auf einer Skala von eins (*trivial*) bis neun (*critical*) ein. Insgesamt kamen die verschiedenen Methoden auf folgende Werte:

- Heuristische Evaluation Rate 3,59
- Usability-Test Rate 4,15
- *Guidelines* Rate 3,61
- *Cognitive Walkthroughs* Rate 3,44

**Reviews finden leichte Fehler, Usability-Tests eher schwere**

Hier läßt sich schon erkennen, daß die Reviews mit Werten um 3,5 eher die ‚leichten Fehler' finden, während über Usability-Tests (Rate über 4,0) eher schwere Fehler gefunden werden. Das Verhältnis von leichten zu schweren Fehlern geht bei den Usability-Tests sehr deutlich in Richtung schwere Fehler. Diesen Sachverhalt verdeutlicht folgende Tabelle:

| | **Heuristic Evaluation** | **Usability** | **Guide-lines** | **Cognitive Walkthrough** |
|---|---|---|---|---|
| **most severe** | 28 | 18 | 12 | 9 |
| **least severe** | 52 | 2 | 11 | 10 |

Tabelle 4: Fehler, die gefunden worden sind, nach Schweregrad (JEFFRIES et al. 1991 S. 122)

**Resümee von Karat, C.-M.**

KARAT, C.-M. resümierte die Ergebnisse der Jeffries-Studie wie folgt (1994 S. 207):

- Heuristische Evaluation findet die meisten Probleme und die meisten der ernsthaften (*serious*) Fehler.
- Usability-Tests finden ernsthafte und häufig wiederkehrende (*recurring*) Fehler, aber keine Konsistenzfehler.
- *Cognitive walkthrough* versagte beim Identifizieren von generellen und wiederkehrenden Fehlern.
- *Guideline walkthrough* findet einige schwere Fehler gar nicht.

**Schnittmenge verschiedener Methoden**

DESURVIRE (1994) fand durch eine Auswertung verschiedener Studien heraus, daß je nach Schwere der gefundenen Probleme die Fehlermenge, die von mehreren Methoden gleichzeitig entdeckt werden (Schnittmenge), variiert. Bei leichten Problemen gibt es zwischen Heuristischer Evaluation und Labortest eine Übereinstimmung von 80 %, während bei schweren Problemen die Übereinstimmung unter 30 % liegt (vgl. DESURVIRE 1994 S. 185). Auch diese Stu-

die zeigt, das Heuristische Evaluation deutlich mehr Fehler findet als das *cognitive walkthrough* und zwar auch bei unterschiedliche Fachkenntnis der ExpertInnen.

Bei vier von Jeffries verglichenen Methoden lag die paarweise Übereinstimmung der gefundenen Fehler jeweils nur bei 10 % bis 15 % (vgl. KARAT C.-M. 1994 S. 207).

### 2.6.3 Verhältnis von Kosten und Nutzen

**Kosten sind nicht direkt vergleichbar**

Die Kosten der verschiedenen Evaluationsverfahren sind nicht direkt miteinander vergleichbar, da einige Methoden sehr technikaufwendig sind. Verglichen werden kann allerdings der Einsatz von menschlicher Arbeit in einem Test. In der Studie von JEFFRIES et al. wurde folgender durchschnittlicher Zeitaufwand für die Evaluation und das Training der Methode festgestellt (1991 S. 122)

- Heuristische Evaluation: 20 Personenstunden
- Usability-Tests: 199 Personenstunden
- *Guidelines walkthrough*: 22 Personenstunden
- *Cognitive walkthrough*: 37 Personenstunden

**Weitere Aufwendungen je nach Methode**

Je nach Methode kommen noch weitere Aufwendungen hinzu, weil auch andere Faktoren eine Rolle spielen: Daten müssen analysiert werden, Berichte müssen geschrieben werden.

Um eine wirkliche Vergleichbarkeit herzustellen, gibt es diverse Methoden. JEFFRIES et al. stellt eine Kosten-/Nutzenanalyse auf, in der die Schwere der gefundenen Fehler und die Zeit, die zum Finden benötigt wurde, den Kosten gegenübergestellt wird. Heuristische Evaluation findet nach dieser Studie im gleichen Zeitraum viermal so viele Fehler wie andere Methoden (1991 S. 122).

**Kosten-/Nutzen durch return of investment**

Karat stellt eine andere Kosten-/Nutzenanalyse auf. Jeder Fehler, der nicht gefunden wird, verursacht Kosten. Wird der Fehler durch eine Evaluation gefunden, so ist dies quasi zurückgewonnenes Kapital (*return of investment*). Es werden die Kosten der Evaluation mit dem zurückgewonnenen Kapital verglichen. Dazu ein Beispiel: Eine Methode kostet 2000 $, findet aber einen signifikanten Fehler nicht, der später korrigiert werden muß und 100.000 $ kostet. Diese Methode ist von der Kosten-/Nutzenanalyse schlechter als eine Methode, die 20.000 $ kostet, diesen Fehler aber findet. Aus diesem Blickwinkel sind Usability-Tests deutlich „billiger“ als andere Methoden, weil mehrheitlich die die schwerwiegenden Fehler gefunden werden (vgl. KARAT C.-M. 1994 S. 219).

### 2.6.4 Resümee

**Reviews in frühen Phasen des Entwicklungszyklus**

Insgesamt ist anerkannt, das *inspection methods* insbesondere in sehr frühen Phasen des Software-Entwicklungszyklus eingesetzt werden sollten (vgl. KARAT C.-M., CAMPBELL & FIEGER 1992, KARAT C.-M. 1994, BROOKS 1994). Einige Methoden, wie *cognitive walkthroughs,* können schon im Designstadium ohne funktionierende Prototypen angewandt werden (vgl. RIEMAN, FRANZKE &

REDMILES 1995 S. 387). BROOKS schreibt dazu: *In summary, inspection methods can be used effectively when the objectives of the study are to identify usability problems and choose from among the design alternatives in the early stages of the product development cycle.* (1994 S. 257).

**Heuristische Evaluation ist effektiv und effizient**

Von den verschiedenen *inspection methods* findet Heuristische Evaluation die meisten Probleme. Obwohl die Methode sehr effektiv ist, ist der Kostenaufwand bei der Heuristischen Evaluation am geringsten (vgl. JEFFRIES et al. 1991). Mit Heuristischer Evaluation verfügen wir über eine Methode, die sowohl effektiv als auch effizient ist. Sie kann während des gesamten Software-Entwicklungszyklus angewandt werden.

**Usability-Test ist teuer**

Mit Usability-Tests verfügen wir ebenfalls über eine Methode, die sehr effektiv ist. Diese Methode findet vor allem schwerwiegende *usability problems* und Probleme, die sehr stark mit dem Ablauf einer Arbeitsaufgabe zusammenhängen. Diese Methode ist aber nicht sehr effizient, weil ein relativ großes Testequipment erforderlich und die Auswertung der Daten sehr zeitaufwendig ist.

**Beide Methoden sind wichtig**

Beides, Usablitity-Tests und *inspection methods,* muß durchgeführt werden. NIELSEN rät, zunächst eine Heuristische Evaluation durchzuführen, um die Benutzungsoberfläche zu säubern (*clean up*) und die offensichtlichen Mängel zu beseitigen. Nachdem die Benutzungsoberfläche redesignt wurde, sollte ein Usability-Test durchgeführt werden (1993 S. 225). Was in diesem Zusammenhang fehlt und benötigt wird, ist eine effiziente Methode, Usability-Tests durchzuführen, die ebenfalls wie die Reviewmethoden während des gesamten Entwicklungszyklus der Software angewandt werden kann.

## 2.7 Discount Usability-Engineering

**Nachteile der bisherigen Methoden**

In den vorherigen Kapiteln haben wir verschiedene Methoden und Verfahren vorgestellt, um Usability von Softwareprodukten zu evaluieren und damit letztendlich zu verbessern. Alle Methoden haben ihre Vor- und Nachteile, die sorgfältig gegeneinander abgewogen werden müssen, damit man für die Erfüllung bestimmter Usability-Ziele die Methode optimal auswählen und einsetzen kann. Hauptnachteile sind (vgl. NIELSEN 1993 S. 17, NIELSEN, GRAY et al. 1998):

- Hoher Aufwand
- Hohe Kosten
- Enormer Zeitbedarf
- Fehlendes Verständnis bei EntwicklerInnen und Software-ManagerInnen

**Discount Usability-Engineering als Lösungsmöglichkeit**

NIELSEN entwickelte Anfang der neunziger Jahre ein Verfahren, um eben diese Nachteile zu beheben. Er geht davon aus, daß es vorteilhafter ist, simple und verständliche Methoden einzusetzen, die allerdings nicht die Qualität von komplexen Verfahren erreichen. Sie werden jedoch eher ausgewählt und tatsächlich eingesetzt. Gleichzeitig erfordern sie wenig Aufwand und verursachen dadurch geringere Kosten (1993 S. 17).

**Discount Usability als Vorbereitung und nicht als Unterdrückung anderer Methoden**

*Discount Usability-Engineering* dient auf der einen Seite dazu, konkrete Usability-Probleme ohne großen Aufwand zu erkennen und beseitigen, während auf der anderen Seite gleichzeitig das Verständnis für die Usability-Problematik bei den EntwicklerInnen und ManagerInnen vertieft werden soll. Reicht die Qualität des *Discount Usability-Engineering* nicht (mehr) aus, so besitzt man immerhin schon so viel Vorwissen, parallel oder zusätzlich aufwendigere und qualitativ höherwertige Verfahren einzusetzen.

**Definition von Discount Usability-Engineering**

Dieses vorgestellte Verfahren, *Discount Usability-Engineering*, besteht aus einer Kombination der folgenden Methoden (vgl. NIELSEN 1993 S. 17), die teilweise in den vorigen Kapiteln vorgestellt wurden:

- BenutzerInnen- und Aufgabenbeobachtung (*user and task observation*)
- Szenarien (*scenarios*)
- Vereinfachtes Lautes Denken (*simplified thinking aloud*)
- Heuristische Evaluation (*heuristic evaluation*)

**Verzicht auf Videodokumentation**

Grundlegend für den Einsatz von *Discount Usability* ist der direkte Fokus auf die BenutzerInnen. Bei der Aufgabenbeobachtung bzw. -analyse gilt als Grundregel, die BenutzerInnen nur zu beobachten, ohne sie bei ihrer eigentlichen Arbeit zu stören (vgl. NIELSEN 1993 S. 18). Die einzelnen Methoden - bis auf Heuristische Evaluation die in Kapitel 2.4 detailliert beschrieben ist - werden im folgenden näher vorgestellt.

### 2.7.1 Szenarien

**Arbeitsaufgaben im Mittelpunkt**

Bei der *Discount Usability*-Methode werden den BenutzerInnen Arbeitsaufgaben vorgelegt, die sie unter Beobachtung abarbeiten müssen. Diese Aufgaben müssen jedoch nicht auf fertigen System-Oberflächen basieren, die komplett implementiert wurden.

**Szenarien als Prototypen**

Der Einsatz der Szenario-Technik anstelle von voll funktionsfähigen Prototypen oder Systemteilen reduziert die Komplexität und damit die Kosten des Tests.

Szenarien stellen eine vereinfachte Form des Prototyping dar. Aus den horizontalen (Anzahl der Funktionalitäten) und vertikalen (Ausprägung der Funktionalität) Prototypen wird eine Schnittmenge als Szenario ausgewählt und mit einer Aufgabe kombiniert den BenutzerInnen präsentiert (vgl. NIELSEN):

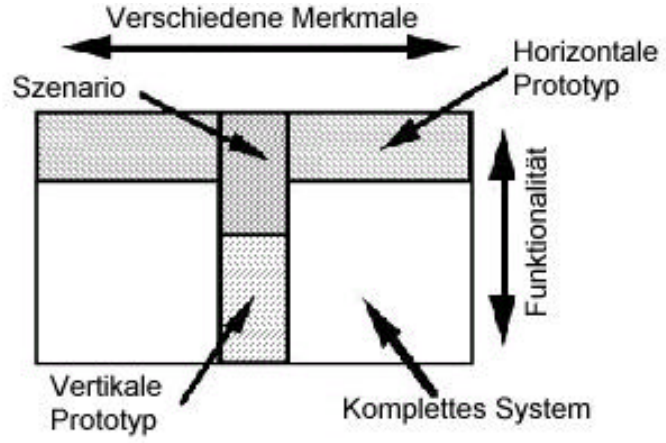



Abbildung 6: Szenario als Prototyp

**Szenarien vom Papiermodell bis zum kompletten System**

Das Szenario kann genau auf das Ziel des Tests hin konzipiert und gewählt werden. Die Form des Szenarios ist undefiniert. Sie kann zwischen einem Papiermodell und dem komplett implementierten System variieren (vgl. NIELSEN 1993 S. 18, KARAT et al. 1992, NARDI 1992).

Es sei darauf hingewiesen, daß der Begriff „Szenario" und seine semantische Bedeutung nicht einheitlich festgelegt ist. Im Bereich der HCI wurde 1992 über die verschiedenen Ansichten für den Bereich der Software-Ergonomie diskutiert (vgl. KYNG 1992, YOUNG & BARNARD 1992, WRIGHT 1992, NARDI 1992, REISNER 1992, CAMPBELL 1992). KARAT J. & KARAT C.-M. schlugen beispielsweise die folgenden vier Einsatzgebiete für Szenarien vor:

- „*Illustrations of use*
- *sample tasks for usability testing*
- *sample task to guide design*
- *test cases for HCI theory*" (1992 S. 4).

Da wir in dieser Arbeit Szenarien in verschiedenen Kontexten benutzen, wird die genaue Semantik bei jeder Erwähnung näher erläutert.

### 2.7.2 Vereinfachtes Lautes Denken

**Vereinfachung der bisherigen Technik**

Das vereinfachte Laute Denken basiert auf den gleichen Inhalten wie die bereits vorgestellte Technik des Lauten Denkens. Die Hauptunterschiede liegen in der Erfahrung und Ausbildung der TestbegleiterInnen und der Methode zur Aufzeichnung der Äußerungen der BenutzerInnen.

**Verzicht auf ExpertInnen**

Bei der traditionellen Methode mußten die BegleiterInnen gründliches Fachwissen über diese Technik besitzen (PsychologInnen usw.), und die Daten des Tests wurden aufwendig aufgezeichnet (Video). Im *Discount Usability-Test* wird auf den Einsatz der ExpertInnen und einer kompletten Dokumentation verzichtet.

InformatikerInnen sind nach einer kurzen Einführung in diese Technik bereits in der Lage, diese Methode einzusetzen. Die Analyse des Tests basiert auf den Notizen, die sie während des Tests anfertigen. Laut Nielsen reichen diese Vor-

aussetzungen aus, einen großen Anteil der Usability-Probleme erkennen zu können (vgl. NIELSEN 1993 S. 18f, NIELSEN S. 5).

### 2.7.3 Kosten/Nutzen von Discount Usability-Engineering

In verschiedenen Studien wurde nachgewiesen, daß der Einsatz der *Discount Usability*-Methoden einerseits die Kosten effektiv senken kann, während andererseits die Qualität des Ergebnisses ausreichend ist (vgl. NIELSEN).

Nielsen gibt an, daß die Kosten für einen Usability-Test fast halbiert werden können, wenn die Methode des *Discount Usability-Engineering* angewandt wird. Folgende Tabelle verdeutlicht die einzelnen Einsparungspotentiale:

| | |
|---|---|
| Geschätzte Usability-Kosten bei herkömmlichen Methoden | $ 128.300 |
| Szenario als Papiermodell statt Videoaufzeichnung | -$ 2.160 |
| Prototyping mit freiem Programmpaket | -$ 16.000 |
| Drei statt fünf BenutzerInnen für die Tests | -$ 11.520 |
| Vereinfachtes Lautes Denken | -$ 5.520 |
| Verzicht auf Video-Labor | -$ 17.600 |
| Zwei statt drei Gruppen für Marktforschung | -$ 2.000 |
| Eine statt drei Gruppen für Akzeptanzanalyse | -$ 4.000 |
| Fragebogen nicht nach jedem Prototyp | -$ 7.200 |
| Usability-Experten für die Heuristische Evaluation | +$ 3.000 |
| Kosten des *Discount Usability-Tests* | $ 65.300 |

Tabelle 5 Kosten eines *Discount Usability-Tests* (NIELSEN S. 2)

Das Verhältnis von Kosten/Nutzen eines *Discount Usability-Tests* in bezug auf die Anzahl der BenutzerInnen verdeutlicht folgende Abbildung:

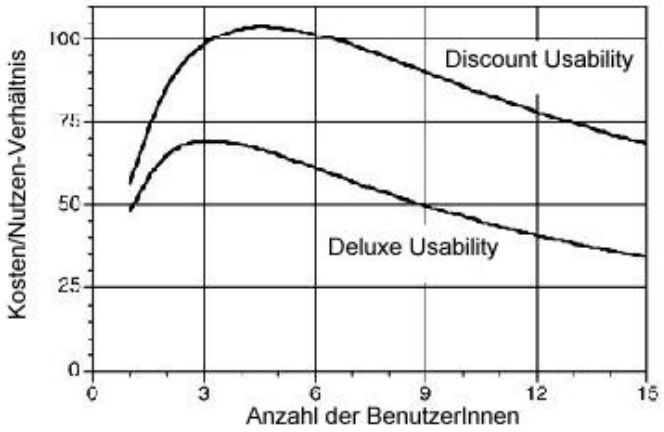



Abbildung 7: Kosten/Nutzen von *Discount Usability-Engineering* (NIELSEN S. 5)

### 2.7.4 Diskussion über Discount Usability-Engineering

**Bisherige Methoden sind zu komplex**

Dem Grundansatz für *Discount Usability-Engineering* liegt die Erkenntnis zugrunde, daß herkömmliche Usability-Methoden nicht oder nur selten angewendet werden, weil ihr Einsatz zu komplex ist und von den Betroffenen nicht verstanden wird: „*Laboratory usability studies cost too much and take too long. As a result, they are rarely done.*" (Atwood in GRAY et al. 1998)

NIELSEN hat mit seinem Vorschlag, auf umfassende und vielschichtige Methoden zu verzichten, eine Methode entworfen, die ohne großen Aufwand direkt eingesetzt werden kann: *(1) it smoothes the way by lowering the threshold of getting started, and (2) it can be used on fast-paced or low-budget projects even in organizations that use a more careful approach for their high-priority projects.*" (NIELSEN in GRAY et al. 1998). Der Einsatz dieser Methoden erreicht auf der einen Seite, daß die betrachtete Software in ihrer Benutzbarkeit verbessert wird, während auf der anderen Seite Bewußtsein und Wissen von der Problematik der Usability bei den Beteiligten erweitert wird.

**Discount Usability ist kein Patentrezept**

Bei der Entscheidung für den Einsatz von *Discount Usability-Methoden* muß jedoch beachtet werden, daß diese Methoden nicht alle Probleme beseitigen können: „*Discount methods will do us a disservice only if we again seek general panaceas, and fail to recognize that the problems we address are diverse, indeed open-ended, and that our methods need to match this diversity.*" (Caroll in GRAY et al. 1998).

Falls die Ziele des Usability-Tests mit den Techniken des *Discount Usability*-Ansatzes nicht erreicht werden können, müssen verschiedene Methoden miteinander kombiniert werden: „*For discount methods, as for all methods, we need to enumerate what evaluation goals are served and how, the costs, the benefits, the conditions of application.*" (Caroll in GRAY et al. 1998).

## 2.8 Ergonomische Ad-hoc-Tests

**Evaluation aus dem Augenblick heraus**

Evaluation allgemein muß während des gesamten Software-Entwicklungszyklus stattfinden. Je früher ein Fehler gefunden wird, desto höher ist das *return of investment*. Es gibt weitere Gründe, warum es sinnvoll sein kann, aus dem Augenblick heraus Feedback von ExpertInnen und BenutzerInnen zu erhalten, beispielsweise:

- bei Designentscheidungen
- beim Bewertung von Alternativen
- bei Streitigkeiten im Designteam, z. B. bei Styleguidebewertungen
- für kurze Tests von fertigen Dialogsequenzen.

**Evaluation eigens zu diesem Zweck**

Es ist sinnvoll, eigens zu diesem Zweck Evaluationen im frühen Stadium durchzuführen. Die Attribute „aus dem Augenblick heraus" und „eigens zu diesem Zweck" führen uns zu dem Begriff Ad-hoc.[43]

**Definition: Ergonomische Ad-hoc-Tests**

Der Begriff ergonomische Ad-hoc-Tests ist in der Literatur wenig geläufig, so daß wir ihn für diese Arbeit wie folgt definieren wollen:

Ergonomische Ad-hoc-Tests sind Evaluationsmethoden, die mit geringem Personal- und Zeitbudget sowie niedrigem organisatorischen Aufwand in der Lage sind, Teilbereiche einer Benutzungsoberfläche zu bewerten. Ad-hoc-Tests können expertInnengestützt (Heuristische Evaluation) und benutzerInnengestützt (Ad-Hoc Usability-Tests) sein. Die Zeit zwischen dem „Testbedürfnis" und dem Zustandekommen des Tests sollte minimal sein.

**Tests müssen schnell durchgeführt werden**

Um Ad-hoc-Tests genau beschreiben zu können, ist es nötig, verschiedene Faktoren zu betrachten, die wir in diesem Kapitel kurz benennen wollen, um sie im Verlauf der Arbeit zu konkretisieren:

- Organisation
- Personen
- Evaluationsmethoden

### 2.8.1 Organisatorische Voraussetzungen

**Räumliche Entfernung**

Wenn „aus dem Augenblick" heraus Rat gesucht wird, dann muß der Test möglichst sofort durchgeführt werden; die Dauer des Tests spielt hierbei keine Rolle. Softwareentwicklung findet jedoch nur selten als „Inhouse-Entwicklung" statt, bei der EntwicklerInnen, ExpertInnen und spätere BenutzerInnnen räumlich sehr nahe beieinander sind. Eine Zusammenkunft der involvierten Personen ist bei räumlicher Ferne sehr teuer. Es lohnt sich nicht, bei kleinen Problemen und großen Entfernungen, wie zwischen München und Hamburg, zusammenzukommen, um Tests durchzuführen.

**Dokumentation parallel zum Test**

Neben der Vorbereitung und Durchführung der Tests muß eine Auswertung der Testergebnisse stattfinden. Bei klassischen Usability-Methoden ist die Auswertung sehr zeitintensiv. Ein Ad-hoc-Test macht nur dann Sinn, wenn Ergebnisse schnell präsentiert werden. Sinnvoll wäre eine Dokumentation während des Tests durchzuführen, um sie nach dem Test sofort präsentieren zu können.

[43] Ad hoc: Eigens zu diesem Zweck, aus dem Augenblick heraus (MEYERS LEXIKON 1997).

### 2.8.2 Personen

**Alle involvierten Personen sind notwendig**

Um effektiv Tests durchzuführen, sind sowohl ExpertInnen als auch Benutzer-Innen nötig. Auch im Bereich von Ad-hoc-Tests ist ein Methodenmix von Evaluationsmethoden nötig, um umfassend Fehler zu finden: von dem kosmetischen Fehler bis zur Usability-Katastrophe. Das führt uns zu verschiedenen Personengruppen: ExpertIn, BenutzerIn und EntwicklerIn.

**Wissen der beteiligten Personen**

Die verschiedenen Personen müssen folgende spezifische Kenntnisse haben:

- Die ExpertInnen müssen sehr flexibel auf Probleme reagieren können. Sie benötigen vertiefte Kenntnisse in Usability-Tests und Heuristischer Evaluation sowie im Bereich der Kommunikation. Sie müssen in der Lage sein, Probleme schnell zu erfassen und Lösungsmöglichkeiten anzubieten. Sie haben evtl. nicht die Kenntnis, wie das Programm genau funktioniert, jedoch fundiertes Wissen über Standards, Styleguides etc.
- Die BenutzerInnen müssen geübt in der Durchführung von Tests sein. Der eigene Arbeitsablauf muß so gestaltet sein, daß sie sich schnell auf die Testsituation einstellen können. Die BenutzerInnen haben keine Kenntnisse von Styleguides, Standards und von langfristigen Wirkungen software-ergonomischer Mängel. Das kann aber gerade ihre Kompetenz sein, denn sie können unvoreingenommen die Wirkung des Programms benennen und beurteilen.
- Die EntwicklerInnen haben hohen Zeitdruck. Die Funktionalität steht im Vordergrund, Software-Ergonomie ist das „I-Tüpfelchen". Bei langen Tests macht es keinen Sinn, daß EntwicklerInnen beim Test anwesend sind. Bei Ad-hoc-Tests ist die Anwesenheit von EntwicklerInnen jedoch sinnvoll, damit während des Tests eine ExpertIn-EntwicklerIn-Kommunikation stattfinden kann.

### 2.8.3 Methodenwahl

**Eigenschaften von Ad-hoc-Methoden**

Evaluationsmethoden für Ad-hoc-Tests müssen folgende Eigenschaften haben:

- **Verfügbarkeit:**
  Die Methode muß ohne großes und kompliziertes Testequipment verfügbar sein. Das sind einerseits Methoden, die sich auf ExpertInnenwissen stützen, und andererseits Methoden, die über Software realisiert werden können.
- **Effektivität:**
  Die Methoden müssen verschiedene Fehlerklassen abdecken und viele Fehler in möglichst kurzer Zeit finden. Im wesentlichen sollen software-ergonomische Katastrophen gefunden werden.
- **Effizienz:**
  Die Methoden müssen effizient in der Vorbereitung, Durchführung und Auswertung sein.

# 3 Ad-hoc Usability-Tests

**Ad-hoc Usability als eine Methode ergonomischer Ad-hoc-Tests**

Usability-Tests werden im wesentlichen am Ende eines Meilensteins innerhalb des Software-Entwicklungszyklus durchgeführt (vgl. NASA). Bei einigen Software-Entwicklungsmodellen, wie dem Wasserfallmodell, findet der Usability-Test sogar erst ganz zum Schluß statt: „*A problem with this waterfall approach is that there will then be no user interface to test with real users until this last possible moment.*“ (NIELSEN 1993 S. 94). Ad-hoc Usability-Tests sollen hier eine Lücke schließen und die gesamte Softwareentwicklung begleiten.

Da der Ad-hoc Usability-Test eine Teilmenge ergonomischer Ad-hoc-Tests ist, muß auch diese Testart die allgemeinen Eigenschaften von Ad-hoc-Tests besitzen:

- Verfügbarkeit
- Effektivität
- Effizienz

**Ad-hoc Usability-Tests haben ähnliche Eigenschaften wie Tests im Labor**

Wir gehen bei unserer Arbeit von folgender These aus: Ad-hoc Usability-Tests haben ähnliche Eigenschaften wie herkömmliche Tests im Labor oder im Feld. Sie finden vergleichbare Fehler, im wesentlichen software-ergonomische Katastrophen. Diese These muß mit Hilfe von Feldstudien bewiesen werden.

## 3.1 Voraussetzungen für Ad-hoc Usability-Tests

**Inhaltliche Voraussetzungen für Ad-hoc Usability-Tests**

In Kapitel 2.8 haben wir bereits Gründe für die Durchführung von ergonomischen Ad-hoc-Tests genannt. Wir werden im folgenden die inhaltlichen Voraussetzungen darstellen, mit denen Ad-hoc-Tests effektiv und effizient durchgeführt werden können.

### 3.1.1 Vertrauensbasis bei den beteiligten Personen

**Personen müssen bekannt sein**

Voraussetzung für einen Usability-Test ist, daß die beteiligten Personen miteinander bekannt sind. Der Test ist immer ein Eingriff in die Persönlichkeitssphäre der Testpersonen, weil nicht nur software-ergonomische Probleme, sondern auch individuelle Fehler der BenutzerInnen aufgedeckt werden. Solche Fehler sind (vgl. KENSIK, PRÜMPER & FRESE 1995):

**Fehlertaxonomie**

- Wissensfehler
- Denkfehler
- Merk-/Vergessensfehler
- Urteilsfehler
- Gewohnheitsfehler
- Unterlassensfehler
- Erkennensfehler
- Bewegungsfehler

**Vertrauensbasis**

Da einige dieser Fehler sehr stark persönlichkeitsabhängig sind (so haben einige Menschen ein besseres Gedächtnis als andere), muß hier vor allem darauf geachtet werden, daß die Daten des Tests nur zur Evaluation und nicht für andere Zwecke eingesetzt werden, z. B. zur Leistungs- und Verhaltenskontrolle. Die mögliche Zweckentfremdung ist ein Grund dafür, daß eine Vertrauensbasis zwischen BenutzerInnen und den anderen beteiligten Personen im Ad-hoc-Test hergestellt werden muß.[44]

**Kennenlernen außerhalb des Tests**

Usability-Tests, die im Labor oder vor Ort durchgeführt werden, haben deswegen eine obligatorische Vorbereitungsphase, bei der sich BenutzerInnen und das Testteam kennenlernen (vgl. BRANAGHAN). An der Durchführung der herkömmlichen Usability-Tests sind BegleiterInnen beteiligt, die diese Verbindung weiter vertiefen. TestbegleiterInnen sind dagegen bei Ad-hoc-Tests nicht vorgesehen, so daß der Prozeß des Kennenlernens und Vertrauenschaffens vorgeschaltet werden muß. Es ist sinnvoll, daß sich die Personen mindestens einmal getroffen haben, bevor der erste Test durchgeführt wird. Dies spielt eine entscheidende Rolle, weil sich durch den Remote-Charakter von Ad-hoc-Tests die Personen während des Tests gar nicht oder nur rechnervermittelt[45] sehen und miteinander kommunizieren können.

**Vetrauensbasis auch zwischen EntwicklerIn und ExpertIn**

Selbstverständlich muß es auch eine Vertrauensbasis zwischen EntwicklerIn und ExpertIn geben. Die EntwicklerInnen müssen sich sicher sein, daß die Anmerkungen und Kommentare der ExpertInnen fachlich einwandfrei sind, da bei Ad-hoc-Problemen meist kein weiterer Rat eingeholt wird.

### 3.1.2 Motivation der beteiligten Gruppen

**BenutzerInnenbeteiligung**

Für die Entwicklung guter Software ist die Beteiligung von BenutzerInnen im gesamten Prozeß der Softwareentwicklung von der Aufgabenermittlung bis zur Evaluation notwendig. Daraus läßt sich ableiten, daß auch AnwenderInnen[46] von Software ein Interesse an der Benutzerbeteiligung haben. Dieser Ansatz ist der Kern eines Ad-hoc Usability-Tests, denn alle beteiligten Gruppen müssen ein Interesse und demnach auch die Motivation für eine solche Zusammenkunft haben.

**Motivation**

Die Motivation der verschiedenen Personen ist völlig unterschiedlich:

- **AnwenderInnen/BenutzerInnen:**
  Die Software soll benutzbar sein. Für die AnwenderInnen werden die ökonomischen Vorteile[47] gut benutzbarer Software im Vordergrund

[44] Zur Diskussion ethischer Aspekte von Tests vgl. NIELSEN (1993).

[45] Dieses können sein Chat, E-Mail, Voicekonferenz, Videokonferenz, aber auch nur bedingt rechnervermittelte Medien wie das Telefon.

[46] AnwenderInnen sind diejenigen, die den Einsatz von Software verantworten.

[47] HERING (1992) geht davon aus, daß in der Bundesrepublik 1988 etwa 15 % zusätzliche Betriebszeit für Wiederholungsläufe und 8 % wegen Softwarefehler zur Verfügung gestellt werden mußten. Bereits eine Verringerung der Test- und Fehlerzeiten um 10 % würde eine Kosteneinsparung von 460 Mio. DM bedeuten.
KEIL et al. haben ermittelt, daß amerikanische Firmen noch höhere Summen für die Überarbeitung von Software-Projekten ausgeben: „*In 1995, U.S. companies alone spent an estimated $59 billion in cost overruns on IS projects and another $81 billion on canceled software projects.*“ (1998 S. 76).

stehen, für die BenutzerInnen die Reduzierung von Zeit und die Erhöhung der Zufriedenheit.

- **EntwicklerInnen**
  EntwicklerInnen stehen zunehmend unter dem Druck, Software herzustellen, die benutzbar im Sinne der Bildschirmarbeitsplatzverordnung ist. Sie benötigen Ad-hoc-Tests u. a. für Designentscheidungen, Bewertung von Alternativen und der Evaluierung fertiger Dialoge, um Wiederholungsfehler zu minimieren.

- **ExpertInnen**
  ExpertInnen haben das Interesse, neben dem ökonomischen Beratungsinteresse ihre Erkenntnisse in der Software-Ergnomie weiterzuvermitteln. Sie wollen die Kosten der Beratung reduzieren und sind deshalb an günstigen Beratungsmethoden interessiert, insbesondere an geringen Nebenkosten.

Wir werden die Interessen der einzelnen beteiligten Gruppen in Kapitel 4 eingehend diskutieren, um aus ihnen Anforderungen an ein System zur Unterstützung von Ad-hoc Usability-Tests abzuleiten.

**EntwicklerInnen profitieren am meisten vom Ad-hoc-Gedanken**

Gegenüber einer allgemeinen Motivation, Tests durchzuführen, ist die Einsicht in die Notwendigkeit der Tests bei den EntwicklerInnen am höchsten. Wir haben schon mehrfach im Rahmen dieser Arbeit auf die Wichtigkeit von Usability-Tests in den frühen Phasen der Software-Entwicklung hingewiesen. Diese Wichtigkeit übertragen wir auf die Notwendigkeit von Ad-hoc-Tests.

### 3.1.3 Wissen akquirieren

**Systemerläuterungen vor dem ersten Test**

Jeder Test hat eine Vorbereitungs-, eine Durchführungs- und eine Nachbereitungsphase. Bei Ad-hoc-Tests verschwimmen diese klar definierten Phasen. Deshalb muß vor dem ersten Test neben dem Aufbau einer Vertrauensbasis eine Phase der Wissensakquise stattfinden:

- Das zu prüfende System muß von der EntwicklerIn als solches dargestellt werden, damit ExpertInnen und BenutzerInnen sich ein Bild von der Gesamtheit machen können. Des weiteren fehlt im frühen Stadium der Softwareentwicklung noch jegliche Dokumentation.

- Einzelne Arbeitsschritte müssen von den EntwicklerInnen erläutert werden, damit die ExpertInnen bei Bedarf oder auf Vorrat Arbeitsaufgaben entwickeln können.

- ExpertInnen müssen Unterlagen für ein Ad-hoc-Review oder einen Ad-hoc Usability-Test greifbar haben, wie z. B. Styleguides, um schnell Auskünfte in diesem Bereich geben zu können.

**Daten zusammenstellen und archivieren**

Informationen, die eigens zu einem bestimmten Software-Entwicklungsprojekt gehören, müssen in geeigneter Weise zusammengestellt und archiviert werden, damit sie im Bedarfsfall schnell zur Verfügung stehen.

### 3.1.4 Kontaktaufnahme

**Verfügbarkeit von BenutzerInnen und ExpertInnen**

Die vorläufige Definition von Ad-hoc-Tests lautete, daß die Zeit zwischen dem „Testbedürfnis“ und dem Zustandekommen des Tests minimal sein sollte (vgl. Kapitel 2.8). Dies setzt eine ständige Verfügbarkeit von ExpertInnen und BenutzerInnen voraus, die allerdings lediglich bei Inhouse-Entwicklungen sichergestellt sein kann.

**Schritte der Kontaktaufnahme**

Die Kontaktaufnahme wird deshalb in gut organisierten Schritte vor sich gehen müssen:

1. Zunächst muß das Problem spezifiziert werden. Es muß entschieden werden, ob das Problem nur Ad-hoc gelöst werden kann.
   Diese Entscheidung kann nicht immer von einer Person allein gefällt werden. Realistisch ist zunächst die Kontaktaufnahme mit einer anderen Person, bevor alle involvierten Personen zusammenkommen.
2. Die beteiligten Personen müssen nach dem frühesten Zeitpunkt für eine Konferenz befragt werden (z. B. per Telefon oder E-Mail).
3. Alle beteiligten Gruppen müssen sich auf den Test vorbereiten: EntwicklerInnen müssen Fragestellungen formulieren, ExpertInnen müssen Arbeitsaufgaben entwickeln, BenutzerInnen müssen sich in eine Arbeitsaufgabe eindenken und ausreichend Zeit für den Test haben.
4. Mit Hilfe von Tools, die das Zusammenarbeiten fördern (z. B. Chat, *whiteboard*), wird die Arbeitsaufgabe diskutiert und optimiert. Ziel ist es, die Arbeitsaufgabe gemeinsam zu gestalten, um optimale Voraussetzungen für den Test zu schaffen.

Erst wenn diese Rahmenbedingungen gewährleistet sind, kann der Kontakt in einer Konferenz zustande kommen und die Durchführung des Tests beginnen.

## 3.2 Gegenstand und Reichweite von Ad-hoc Usability-Tests

**Reviews vor Usability-Tests**

Um wirkungsvoll Usability-Tests durchzuführen, muß die Software eine bestimmte Güte und einen gewissen Entwicklungsstand erreicht haben. So müssen Reviewmaßnahmen bereits durchgeführt worden sein (vgl. auch NIELSEN 1993). „Bestandene Reviews dienen somit einer (auch formalen) ‚Freigabe zum Usability-Test‘. Ohne Vorschaltung besteht die Gefahr, sich während des aufwendigen Usability-Tests in der Erkennung von Fehlern bei Anordnung und inkonsistenten Bezeichnern o. ä. zu verlieren.“ (ANSORGE & HAUPT 1997 S. 65). Daraus folgt, daß ein bestandenes Ad-hoc-Review (beispielsweise die Heuristische Evaluation) der „Freigabe“ zum Ad-hoc Usability-Test dient. Dieses Review führt ein *clean up* der Benutzungsoberfläche durch. Erst danach können andere Testverfahren greifen. Aus einer solchen Vorgehensweise resultiert, daß sich alle Teilnehmenden auf das Wesentliche, nämlich auf die Durchführung des Tests konzentrieren können. Wir wollen die Methoden des *Discount Usability-Engineering* (vgl. Kapitel 2.7), für einen gesamten Ad-hoc-Test vorschlagen, weil hier sowohl das *clean up* mit Hilfe der Heuristischen Evaluation als auch verschiedene Methoden im Bereich des Usability-Tests unterstützt werden. Wir definieren Ad-hoc Usability-Test deshalb für die weitere Arbeit wie folgt: Ein Ad-hoc Usability-Test ist eine Zusammenfassung von Methoden, mit deren Hilfe schnelle Evaluationen durchgeführt werden kön-

nen. Diese Methoden verstehen sich analog dem *Discount Usability*-Gedanken von NIELSEN:

- Heuristische Evaluation für das *clean up* der Benutzungsoberfläche,
- Szenariotechnik als vereinfachtes Prototyping,
- Vereinfachtes Lautes Denken und
- BenutzerInnen und Aufgabenbeobachtung.

### 3.2.1 Entwicklungsstand der Software

**Ad-hoc-Tests benötigen einen bestimmten Entwicklungsstand**

Die Idee der Ad-hoc-Tests ist die Zeit- und Kostenersparnis während der Entwicklung, indem schon frühzeitig alle Möglichkeiten der Evaluation (Reviews und Usability-Tests) durchgeführt werden können. Dafür muß die zu untersuchende Software einige Eigenschaften haben, d. h. in einem bestimmten Entwicklungsstand sein: „*In order to tie test and evaluation findings to particular system, one need prototypes at some level, so that at least the system concepts can be portrayed in a concrete form.*" (LINDGAARD 1994 S. 91).

**Ad-hoc-Tests benötigen Prototypen**

Ad-hoc-Tests werden mit Prototypen durchgeführt. „Ein Software-Prototyp ist ein – einfach zu änderndes und zu erweiterndes ausführbares Modell des geplanten Software-Produkts, das nicht notwendigerweise alle Eigenschaften des Zielsystems aufweisen muß, jedoch so geartet ist, daß vor der eigentlichen Systemimplementierung der Anwender die wesentlichen Systemeigenschaften erproben kann." (POMBERGER & BLASCHEK 1996 S. 4). Im Kontext mit Ad-hoc-Tests werden im wesentlichen Teile eines Softwaresystems geprüft.

**Im Mittelpunkt steht die Arbeitsaufgabe**

Für die Entscheidung, ob eine Software für einen Ad-hoc Usability-Test „tauglich" ist, stellt sich die Frage: Kann eine repräsentative Arbeitsaufgabe mit dem Prototypen durchgeführt werden?

**Horizontales vs. vertikales Prototyping**

Beim Prototyping wird zwischen horizontalen und vertikalen Prototypen unterschieden (vgl. NIELSEN 1993 S. 94f). Wir wollen dies mit Augenmerk auf eine durchzuführende Arbeitsaufgabe wie folgt erweitern:

- Ein horizontaler Prototyp ist die Simulation einer Benutzungsoberfläche, mit der eine praxisnahe Arbeitsaufgabe nicht durchgeführt werden kann. Mit horizontalen Prototypen kann die gesamte Benutzungsoberfläche auf Konsistenz geprüft werden. Des weiteren kann der Arbeitsablauf begutachtet werden sowie die Schnittstellen zwischen verschiedenen Dialogen.
- Der vertikale Prototyp implementiert eine Funktion des Programms vollständig aus. Es wird ein Ausschnitt des Systems implementiert, und zwar in die Tiefe. Innerhalb des Ausschnitts können kleine Arbeitsaufgaben durchgeführt werden, weil die Funktionalität schon vorhanden ist. Schnittstellen zu anderen Programmteilen können nicht getestet werden.

**Szenario**

Der optimale Prototyp für einen Ad-hoc Usability-Test ist eine Mischung aus vertikalem und horizontalem Prototyp - ein Szenario. Ein Szenario in diesem Kontext ist ein vereinfachter Prototyp, der eine Arbeitsaufgabe mit wenig Fle-

xibilität für die BenutzerInnen beschreibt. NIELSEN (1993 S. 100) definiert ein Szenario wie folgt:

Ein Szenario ist eine Beschreibung:

- von einem individuellen Benutzer,
- der einen bestimmten Ausschnitt der Systemfunktionen benutzt,
- der ein bestimmtes Ziel erreichen will,
- unter spezifischen Verhältnissen,
- innerhalb eines bestimmten Zeitintervalls.

### 3.2.2 Zeitpunkt des Tests

**Es gibt keinen besonderen Zeitpunkt für Tests**

Einen besonderen Zeitpunkt zur Durchführung des Tests gibt es innerhalb des Software-Entwicklungszyklus nicht. Wenn die Software die oben beschriebenen Fähigkeiten hat, kann der Test durchgeführt werden. Die Frage, ob das Problem „reif" für einen Ad-hoc-Test ist, muß sich die EntwicklerIn stellen und beantworten.

**Entscheidung treffen: Das Problem ist Ad-hoc fähig**

Oft haben EntwicklerInnen nicht genug Wissen, um darüber zu entscheiden, ob ein Problem vorhanden ist und wann es notwendig ist, die BenutzerInnen hinzuzuziehen. Deshalb besteht ein Qualifizierungsbedarf bei den EntwicklerInnen:

- **Sensibilisierung für Evaluation:**
  Zunächst müssen die EntwicklerInnen dafür sensibilisiert werden, daß Evaluationen die Güte des Produkts steigern können. Gerade bei der Auswahl verschiedener Alternativen ist es nicht immer so, daß die Auswahl durch die EntwicklerInnen das Bedürfnis der BenutzerInnen trifft. Das Prinzip der BenutzerInnenbeteiligung im gesamten Software-Entwicklungsprozeß muß sich bei den EntwicklerInnen durchsetzen.
- **Kleine Evaluationen (z. B. Ad-hoc-Inspektionen) selbst durchführen:**
  Für bestimmte Probleme im Bereich des *clean up* ist es nicht notwendig, Tests mit externen ExpertInnen durchzuführen. Über Checklisten können EntwicklerInnen befähigt werden, selbst bestimmte Evaluationen vorzunehmen (vgl. KARAT & DAYTON 1995).
- **Einschätzen, ob für die Lösung eines Problems ein Ad-hoc-Test benötigt wird:**
  EntwicklerInnen müssen befähigt werden einzuschätzen, ob das vorliegende Problem mittels eines Ad-hoc Usability-Tests gelöst werden kann. Hier sind unter Umständen Vorkonsultationen mit ExpertInnen nötig. Entscheidend ist dabei: Gibt es eine Arbeitsaufgabe, die eine potentielle BenutzerIn des Systems ausführen kann?

### 3.2.3 Grenzen von Ad-hoc Usability-Tests

Ein Ad-hoc Usability-Test hat bestimmte Grenzen, die hier kurz skizziert werden sollen:

**Zeitdauer nicht definiert**

- Die Zeitdauer eines Tests ist bei Ad-hoc-Tests nicht definiert. Da BenutzerInnen in ihrem Arbeitsumfeld „gestört" werden und die Tests neben ihrem eigentlichen Arbeitspensum durchführen, sollte die Dauer des Tests angemessen sein. Längere Tests müssen unter anderen Rahmenbedingungen durchgeführt werden.

**Inhalte sehr begrenzt**

- Die Inhalte von Ad-hoc-Tests sind sehr begrenzt. Die Tests sind dafür konzipiert, auftretende Probleme bei der Entwicklung von Software schnell zu analysieren und zu lösen. Besondere Aspekte der Evaluation (wie z. B. die Explorationsfreudigkeit einer Software) gehören in das Usability-Labor, weil sie zu aufwendig für die Idee der kleinen Tests sind.

**ExpertInnen brauchen grundlegende Erfahrungen**

- Wie wir in Kapitel 2.4.5 beschrieben haben, steigert fundiertes ExpertInnenwissen die Quote der gefundenen Fehler deutlich. NIELSEN (1992) sprach in diesem Zusammenhang von doppelt-qualifizierten ExpertInnen. Solche ExpertInnen haben nicht nur Erfahrung in der Software-Ergonomie allgemein, sondern auch im Anwendungsgebiet der Software. Daraus ziehen wir den Schluß, daß die eingesetzten ExpertInnen zumindest im Bereich der Software-Ergonomie grundlegende Erfahrungen in der Evaluation von Benutzungsoberflächen besitzen müssen und möglichst das Anwendungsgebiet kennen.

### 3.2.4 Ergebnisse / Dokumentation

**Dokumentation ist sehr zeitaufwendig**

Ein kostenaufwendiger Bestandteil bei herkömmlichen Usability-Tests ist die Auswertung der Ergebnisse und die Erstellung einer Dokumentation. Wenn die Ergebnisse beispielsweise die Qualität eines Gutachtens haben sollen, muß noch mehr Zeit aufgewendet werden. "*A basic problem is that with a few exceptions, published descriptions of usability work normally describe cases where considerable extra efforts were expended on deriving publication-quality results, even though most development needs can be met in much simpler ways.*" (NIELSEN S. 3).

**Schnelle Lösung ist gefordert**

Eine zeitaufwendige Dokumentation widerspricht dem Grundgedanken von Ad-hoc-Tests, daß Probleme aus dem Augenblick heraus entstehen und nach einer schnellen Lösung drängen, besonders wenn diese Probleme eine Weiterarbeit behindern. Die Dokumentation muß dem Problemcharakter angemessen sein. Es gibt Probleme, für die keine Dokumentation erforderlich ist (z. B. bei einer Designentscheidung zwischen zwei Alternativen), und es reicht zur Problemlösung ein Hinweis, der nur kurz begründet werden muß. Bei einem Test treten jedoch mehrere Fehler auf, die einzeln aufgezeichnet werden müssen, damit sie später berichtigt werden können.

**Dokumentation während des Tests**

Sinnvoll wäre eine Dokumentation parallel zum Test. HAMMONTREE, WEILER & NAVAK (1994) schlagen vor, Screenshots in ein *whiteboard* zu kopieren, damit über Zeichnungen und Kommentare die Probleme festgehalten werden können. Eine solche Dokumentation während des Tests kann sofort nach

Beendigung der Arbeitsaufgabe allen Teilnehmenden zugesandt werden und Grundlage der Auswertung sein. Die Kommentierung könnte ebenfalls nach dem Test von allen Teilnehmenden durchgeführt werden.

## 3.3 Durchführung eines Ad-hoc Usability-Tests

**Prozeßschritte des Ad-hoc Usability-Tests**

Wie in Kapitel 2.5.1 beschrieben, lassen sich die einzelnen Schritte eines Usability-Tests gemäß ihrer Aufgabe für den gesamten Test einteilen. Da sich dieses Verfahren in der Praxis durchgesetzt hat, übertragen wir die Schritte des Prozesses auf den Ad-hoc Usability-Test. Wir überprüfen im einzelnen, ob alle Stufen durchführbar sind und welche Änderungen sich gegenüber herkömmlichen Usability-Tests ergeben.

**Informeller Charakter von Ad-hoc Usability-Tests**

Grundsätzlich ist anzumerken, daß Ad-hoc Usability-Tests informeller und weniger aufwendig als normale Usability-Tests geplant und durchgeführt werden können:

1. **Spezifikation des Ziels eines Tests**

   **Zielfestlegung im Zusammenhang mit der Formulierung der Arbeitsaufgabe**

   Das Ziel eines Ad-hoc Usability-Tests wird vor dem Testbeginn im Zusammenhang mit der Formulierung der durchzuführenden Arbeitsaufgabe festgelegt. Dieses geschieht in der Phase der Kontaktaufnahme (vgl. Kapitel 3.1.4). Somit haben alle Beteiligten die Möglichkeit, das Ziel des Tests mitzubestimmen bzw. zu beeinflussen, und es wird auf eine Dokumentation (z. B. in Form eines Fragenkatalogs) verzichtet.

   **Keine explizite Methodenfestlegung**

   Im Gegensatz zu herkömmlichen Usability-Tests ergeben sich aus dem Testziel keine entscheidenden Auswirkungen auf die einzusetzenden Methoden, sondern sie lassen sich direkt aus der Arbeitsaufgabe herleiten.

2. **Bestimmung des BenutzerInnenprofils**

   **BenutzerInnenprofil überflüssig**

   Die Rolle der BenutzerInnen muß in einem Ad-hoc Usability-Test von den realen BenutzerInnen übernommen werden. Aufgrund des Verzichts auf komplexes und aufwendiges Equipment ist es möglich, die betroffenen Personen in die Testdurchführung zu involvieren. Der Test wird praktisch zu den BenutzerInnen gebracht (vgl. ROWLEY 1994). Somit kann auf die Erstellung eines Profils verzichtet werden.

   **Ersatz für echte BenutzerInnen**

   Sollten keine „echten" BenutzerInnen am Test teilnehmen können, so muß eine adäquate Ersatzperson diese Rolle übernehmen. Auf eine fundierte Analyse der „echten" BenutzerInnen-Eigenschaften muß allerdings verzichtet werden, der Aufwand wäre zu groß und würde den Ad-hoc-Charakter des Tests sprengen.

3. **Ermittlung der Anzahl der BenutzerInnen**

   **Analogie von Discount Usability und Ad-hoc Usability**

   Die Anzahl der teilnehmenden BenutzerInnen spielt auch bei Ad-hoc Usability-Tests eine wichtige Rolle. Aus den Studien und Analysen von NIELSEN (1993) läßt sich schließen, daß die Anzahl der gefundenen Usability-Probleme auch bei Ad-hoc Usability-Tests mit der Anzahl der BenutzerInnen steigt, da sich die Ad-hoc Usability-Methoden auf denen der *Discount Usability* stützen. Ob jedoch das Verhältnis von ge-

fundenen Usability-Problemen und Anzahl der BenutzerInnen gleich ist, muß in weiteren Feldstudien untersucht werden.

4. **Ermittlung von Aufgaben**

**Arbeitsaufgaben bei der Kontaktaufnahme mit zentraler Rolle der ExpertInnen**

Im Gegensatz zu den beschriebenen Usability-Tests gestaltet sich die Ermittlung der durchzuführenden Arbeitsaufgaben bei einem Ad-hoc Usability-Test einfacher. Wie schon beschrieben (vgl. Kapitel 3.1.4), werden die Arbeitsaufgaben in der Phase der Kontaktaufnahme von allen Beteiligten mitbestimmt. Dabei ist zu beachten, daß die ExpertInnen eine zentrale Rolle übernehmen, da sie aus ihrem Wissen und ihrer Erfahrung heraus am qualifiziertesten entscheiden können, welche Aufgaben für den jeweiligen Kontext geeignet sind.

**Beibehaltung der Aufgabenbeschreibungen**

Die Arbeitsaufgaben werden analog zu den bisherigen Aufgabenbeschreibungen erstellt und allen Beteiligten übergeben.

5. **Bestimmung von Leistungs- und Zufriedenheitsmetriken**

**Keine expliziten Metriken notwendig**

Für eine Auswertung der Erfüllung der Arbeitsaufgaben werden in normalen Tests Metriken erstellt. Bei Ad-hoc Usability-Tests entfällt dieser Schritt bzw. geht direkt in der Rolle der ExpertInnen auf. Sie entscheiden bei der Beobachtung und Durchführung des Tests, welche Metriken anzuwenden sind oder wann eine Arbeitsaufgabe als erfüllt betrachtet werden kann. Da der zentrale Gegenstand von Ad-hoc Usability-Tests die Erkennung von „Usability-Katastrophen" ist, werden komplizierte und aufwendige Metriken zur Analyse nicht benötigt.

6. **Spezifikation der Rolle der TestmitgliederInnen**

**Bestimmung der Rollen bei der Kontaktaufnahme**

Die Rollen der einzelnen TestmitgliederInnen ist bei Ad-hoc Usability-Tests von vornherein festgelegt. Außer den ExpertInnen, BenutzerInnen und EntwicklerInnen nehmen keine weiteren Personen am Test teil, so daß die Festlegung der Rollen bei der Kontaktaufnahme (vgl. Kapitel 3.1.4) diskutiert und festgehalten werden kann.

7. **Bestimmung des Test-Einrichtungslayouts**

**Keine Layoutbestimmung notwendig**

Auf die Bestimmung des Layout der Testeinrichtung kann verzichtet werden, da aufgrund des einfachen und unkomplizierten Testaufbaus auf diesen Schritt ohne Auswirkung auf die Ergebnisse verzichtet werden kann.

8. **Durchführung eines Pilot-Tests**

**Verzicht auf Pilot-Tests, da einfache Wiederholung möglich ist**

Auf die Durchführung eines Pilot-Tests kann verzichtet werden, da dies dem Ad-hoc-Charakter zuwiderläuft. Sollte der Test nicht die gewünschten Ergebnisse liefern, kann er ohne großen Aufwand mit geänderten Attributen (z. B. andere Arbeitsaufgabe, andere BenutzerInnen) fortgesetzt oder wiederholt werden.

9. **Durchführung des Tests**

**Unkomplizierte und informelle Durchführung**

Entsprechend den Eigenschaften von Ad-hoc Usability-Tests gestaltet sich die eigentliche Durchführung des Tests informell, d. h. ohne daß auf besondere Formalitäten geachtet werden muß. Die TeilnehmerInnen sind sich aufgrund der Kontaktaufnahme ihrer Aufgaben und Rollen während des Tests bewußt. Der Test kann ohne weitere Schritte begonnen und durchgeführt werden.

**Besprechung von Problemen während des Tests**

Sollten während des Tests Unklarheiten oder Probleme auftauchen, können sie während der Durchführung besprochen und geklärt werden. Ist keine Klärung möglich, wird der Test unterbrochen und zu einem späteren Zeitpunkt wiederholt.

**Analyse und Dokumentation der Daten während des Tests**

10. **Analyse der Daten**
    Eine explizite Analyse und Dokumentation der erfaßten Daten als eigenständige Phase entfällt bei einem Ad-hoc Usability-Test, da diese Aufgaben während des Tests von den ExpertInnen übernommen und erledigt werden. Sie erkennen die jeweiligen Usability-Probleme und dokumentieren sie gleichzeitig. Nach Beendigung des Tests stehen alle Daten zur Verfügung und können den Initiatoren des Tests übergeben werden.

**Zusammenfassung zu zwei Schritten**

Zusammenfassend läßt sich nach dieser Analyse schließen, daß für die Durchführung eines Ad-hoc Usability-Test weniger Schritte benötigt werden als bei herkömmlichen Tests. Die folgenden zwei sind ausreichend, um einen vollständigen Ad-hoc Usability-Test zu beschreiben:

1. **Kontaktaufnahme**
   Wichtig ist, daß vor dieser Phase bereits eine Heuristische Evaluation, als Ad-hoc-Review durchgeführt wurde. In dieser Phase werden alle notwendigen Vorbereitungen für die Durchführung des Tests getroffen. Hierzu gehören:
   - Bestimmung des Testziels und Formulierung der Arbeitsaufgabe.
   - Besprechung und Festlegung der Rollen während des Tests.
2. **Testdurchführung**
   Bearbeitung der Arbeitsaufgaben, während gleichzeitig Daten erfaßt, analysiert und dokumentiert werden.

**Optionaler dritter Schritt**

Eine Besprechungs- und Diskussionsphase als dritter Schritt kann ohne Auswirkungen auf die eigentliche Testdurchführung angeschlossen werden.

## 3.4 Einsatzszenarien

**Szenariotechnik als Problemlösung**

Die nachfolgenden Szenarien dienen als Strategieentwürfe. Einerseits wird das Zusammenspiel der Personenkreise analysiert, andererseits werden die Szenarien in den Ablauf eines potentiellen Ad-hoc Usability-Tests eingeordnet. Anschließend analysieren wir die Szenarien daraufhin, ob für die jeweilige Situation ein Ad-hoc Usability-Test angemessen ist.

Grundlage für dieses Herangehen ist die Szenariotechnik, die als „eine Anleitung zur prozeßorientierten Problemlösung“ (SCHOLLES 1998 S. 2) eingesetzt wird. Darüber hinaus wird diese Technik eingesetzt, „*to provide an explicit concrete vision of how some human activity could be supported by technology.*“ (NARDI 1992 S. 13).

**Szenarien während des Ad-hoc Usability-Tests**

Die hier beschriebenen Szenarien grenzen sich grundlegend von den während der Ad-hoc Usability-Tests benutzten Szenarien ab. Für die Durchführung des Usability-Tests werden Arbeitsaufgaben benötigt, die Aspekte der Software verdeutlichen sollen und mittels Szenarien beschrieben werden (vgl. Kapitel 2.7).

**Unterschiedliche Personenkreise in den Szenarien**

Für die Darstellung des Ad-hoc Usability-Tests hingegen bedarf es einer anderen Beschreibung. Gemäß der oben aufgeführten Definition für Szenarien beschreiben wir im folgenden das Zusammenspiel der Personenkreise, die an einem Ad-hoc Usability-Test teilnehmen. Wie wir in Kapitel 2.8.2 erarbeitet haben, nehmen an diesem Test die ExpertInnen, BenutzerInnen und EntwicklerInnen teil, wobei alle Beteiligten unterschiedlich motiviert sind und differenzierte Ziele verfolgen (vgl. Kapitel 3.1). Anhand der Szenarien erarbeiten wir, ob immer alle Personen an den Tests teilnehmen müssen oder ob er auch in kleineren Kreisen durchgeführt werden kann.

**Zwei- oder Dreipunkt-Kommunikation**

Diese Unterscheidung hat weitreichende Auswirkungen auf die Vorbereitung und Durchführung der Ad-hoc Usability-Tests. In Kapitel 2.8 haben wir darauf hingewiesen, daß sich die teilnehmenden Personen an unterschiedlichen, in der Regel weit entfernten Orten befinden. Es bedarf organisatorischer Vorbereitungen, die mit der Anzahl der Teilnehmenden zunimmt, um zur Durchführung eines Tests zusammenzukommen. Nehmen nur zwei Personenkreise (z. B. ExpertInnen und EntwicklerInnen) teil, so handelt es sich um eine direkte Kommunikation zwischen beiden (Zwei-Punkt-Kommunikation). Kommen die BenutzerInnen hinzu, so erhöht sich die Komplexität der Verständigung (Dreipunkt-Kommunikation).

**Aufbau der Szenarien**

Die folgenden Situationsbeschreibungen erläutert die äußeren Umstände, die für den Test gelten bzw. zu ihm führen. Wir stellen den Gegenstand des Tests in Form einer Problembeschreibung dar. Durchführung und Arbeitsaufgabe führen zum konkreten Ablauf. Abschließend beschreiben wir die aus dem Test gewonnenen Ergebnisse.

### 3.4.1 EntwicklerInneninitiiertes Szenario

**Situationsbeschreibung:** Die EntwicklerIn erstellt ein datenbank-basiertes System zur Erstellung und Berechnung von Gebühren. Das eigentliche System besteht aus mehreren Komponenten, die folgenden Inhalt haben:

- Datenbanksystem zur Speicherung aller benötigten Strukturen und Daten,
- Oberfläche zur Erstellung und Berechnung von Gebührensätzen,
- Oberfläche zur Erstellung von Rechnungen, die auf den erstellten Gebührensätzen basieren.

Die Komponenten sind vollständig erstellt und in ihrem Ablauf funktional bereits geprüft. Der Auftrag für diesen Test wurde im Vorfeld erteilt, so daß die ExpertIn der detaillierte Kontext bereits bekannt ist.

**Problem:** Gemäß der Arbeitsaufgabe sollen die Komponenten daraufhin untersucht werden, ob sie software-ergonomisch korrekt gestaltet wurden. Die ExpertIn soll diese Frage beantworten, nachdem die BenutzerIn diese Aufgabe durchgeführt hat.

**Durchführung:** Die EntwicklerIn initiiert die Sitzung und präsentiert den Entwicklungsvorschlag. Die erstellten Komponenten werden an die BenutzerIn weitergegeben. Die ExpertIn erläutert der BenutzerIn die Arbeitsaufgabe und beobachtet deren Bearbeitung.

**Arbeitsaufgabe:** Es sollen drei neue Gebührensätze anhand einer geänderten, gedruckt vorliegenden Gebührenordnung erstellt und in der Datenbank gespeichert werden. Anschließend sollen neue Rechnungen erstellt werden, die auf neu berechneten Gebührensätzen basieren.

**Ablauf:**

1. Die ExpertIn führt eine Heuristische Evaluation durch, um die gröbsten ergonomischen Verstöße zu erkennen.
2. Die EntwicklerIn bereinigt die Fehler, die in der Heuristischen Evaluation gefunden wurden.
3. Die Sitzung wird durch die EntwicklerIn initiiert.
4. Die zu testenden Komponenten werden von der EntwicklerIn an die ExpertIn und BenutzerIn übertragen.
5. Die ExpertIn bespricht die Arbeitsaufgabe mit den Beteiligten.
6. Die BenutzerIn führt die Arbeitsaufgabe gemäß der Beschreibung durch.
7. Die ExpertIn reagiert entsprechend den Aktionen, die die BenutzerIn ausführt und analysiert ergonomische Verstöße.
8. Der EntwicklerIn werden die Verstöße mitgeteilt.

**Ergebnis:** Das Ergebnis der Sitzung besteht aus einer Sammlung von Hardcopies, die die entsprechenden ergonomischen Verstöße darstellen. Darüber hinaus ist jeder Hardcopy eine Erläuterung des jeweiligen Verstoßes beigefügt.

### 3.4.2 ExpertInneninitiiertes Szenario

**Situationsbeschreibung:** Es existiert ein firmeneigener Styleguide für die Erstellung von Software, der insbesondere die ergonomischen Kriterien festlegt. Die EntwicklerIn hat ein Modul erstellt und freigegeben, das die funktionalen Tests bereits erfolgreich durchlaufen hat. Die ExpertIn überprüft, ob die Implementierung den Richtlinien im Styleguide entspricht.

**Problem:** Eine bestimmte Implementierung ist nicht styleguidegerecht, enthält aber sehr gute Ansätze, die eine Überarbeitung des alten Styleguides anregen. Die ExpertIn will die Meinung der BenutzerInnen einholen bzw. überprüfen, ob die neue Idee praktikabel ist.

**Durchführung:** Die ExpertIn initiiert eine Sitzung. Die neue Lösung wird präsentiert. Die EntwicklerIn erklärt die Beweggründe, die zu einer Änderung geführt haben. Es wird ein Usability-Test durchgeführt.

**Arbeitsaufgabe:** Es liegt keine konkrete Arbeitsaufgabe vor, da eine Oberfläche gemäß eines spezifizierten Styleguides diskutiert wird.

**Ablauf:**

1. Die Sitzung wird initiiert.
2. Die EntwicklerIn startet sein Rapid-Prototyp-System (z. B. Borland Delphi) mit dem zu überprüfenden Dialog und erläutert die Entwurfsentscheidungen.
3. Die ExpertIn erläutert über Screenshots den styleguidegerechten Dialog bzw. die Abweichungen vom Styleguide.
4. Die BenutzerIn kommentiert die verschiedenen Entwürfe.

**Ergebnis:** Das Ergebnis ist eine Bewertungsgrundlage für eine Änderung des Styleguides, die die ExpertIn aus den Anmerkungen und dem bisherigen Styleguide erstellt.

### 3.4.3 BenutzerInneninitiierte Szenario

**Situationsbeschreibung:** Die Benutzerin testet ein freigegebenes Modul oder arbeitet mit einer bestimmten Software.

**Problem:** Während der Arbeit treten Fehler auf, die nur schwer verbal beschreibbar sind. In Diskussionen mit ExpertInnen stellt sich heraus, daß der Fehler auf anderen Rechnern nicht reproduzierbar ist. Der Fehler ist entweder mit der Arbeitsweise der BenutzerIn oder mit der Rechnerkonfiguration verbunden.

**Durchführung:** Die BenutzerIn initiiert eine Sitzung und führt den Fehler vor. Durch a*pplication sharing* (z. B. des Gerätemanagers) können Systemeingriffe durchgeführt und Systemeinstellungen verdeutlicht werden.

**Arbeitsaufgabe:** Es liegt keine konkrete Arbeitsaufgabe vor, die von einer der beteiligten Personen abzuarbeiten wäre, da die BenutzerIn einen Sachverhalt präsentiert, der Gegenstand der Sitzung ist.

**Ablauf:**

1. Es wird ein virtuelles Gerät (z. B. Netzwerkkarte) auf den IRQ der zweiten Schnittstelle installiert, so daß der Gerätemanager einen Ressourcenkonflikt anzeigt.
2. Es wird ein kleines Programm geschrieben, das aufgrund dieses Ressourcenkonflikts einen Fehler meldet (z. B. „Das Gerät funktioniert nicht“).
3. Die Konferenz wird aufgrund des Fehlers von der BenutzerIn initiiert.
4. Das Programm wird geshared, um einen individuellen Benutzungsfehler durch die BenutzerIn möglichst auszuschließen.
5. Die BenutzerIn schickt das ausführbare Programm über den Dateikanal, um es auf dem ExpertenInnen/EntwicklerInnen-Rechner zu überprüfen. Hier tritt der Fehler nicht auf.
6. Die BenutzerIn shared den Gerätemanager des Systems.
7. Die ExpertIn erkennt einen Ressourcenkonflikt in Zusammenarbeit mit der EntwicklerIn und behebt diesen Fehler.

**Ergebnis:** Das Ergebnis ist ein Fehlerprotokoll, das als Grundlage für die Berichtigung des System-Handbuchs dienen kann.

### 3.4.4 Zusammenfassung und Resümee

Aus der folgenden Zusammenfassung der verschiedenen Szenarien ergeben sich Schlußfolgerungen, die einerseits unsere Annahmen untermauern (vgl. Kapitel 3.3) und aus denen sich anderseits neue Anforderungen an ein konkretes System zur Unterstützung von Ad-hoc Usability-Tests lassen (vgl. Kapitel 4) bzw. andere Einsatzmöglichkeiten eines solchen Systems ableiten lassen:

- **Verschiedene Konfigurationen der Sitzungen**

**Bedürfnis für alle Beteiligten**

Bei den drei beschriebenen Szenarien wird der Test jeweils von einem anderen Personenkreis initiiert (EntwicklerIn, ExpertIn, BenutzerIn).

Bei allen Beteiligten kann ein Bedürfnis nach einem Ad-hoc-Test bestehen, ihre Ziele unterschieden sich jedoch grundlegend.

**Dreipunkt-Kommunikation notwendig**

**Ad-hoc Usability-Test**

Die EntwicklerIn benötigt für den präsentierten Entwicklungsvorschlag im ersten Szenario konkrete ExpertInnen-Hilfe und die aktive Beteiligung der BenutzerInnen. Damit ist eine Dreipunkt-Kommunikation erforderlich, deren Informationsströme alle Beteiligten gleichberechtigt berücksichtigen müssen. Es wird ein Ad-hoc Usability-Test durchgeführt, wie er in Kapitel 3 vorgeschlagen und definiert wird.

**Zweipunkt-Kommunikation möglich**

**Ad-hoc Reviews; Präsentation und Diskussion**

Das zweite, expertInneninitiierte Szenario hingegen stellt keinen eigentlichen Ad-hoc Usability-Test dar. In diesem Szenario, in das ebenfalls alle Beteiligten involviert sind, wird zwar auch eine Dreipunkt-Kommunikation aufgebaut, die Kommunikationsströme sind allerdings nicht gleichberechtigt auf alle Beteiligten verteilt. Die InitiatorIn dieser Sitzung präsentiert die gewonnenen Einschätzungen des Sachverhalts (z. B. anhand einer Ad-hoc-Inspektion). Die BenutzerInnen und EntwicklerInnen nehmen eine passive Rolle ein, und können „nur" Kommentare und Bemerkungen in die Kommunikation einbringen. Somit wäre es auch möglich, diese Sitzung über getrennte Zweipunkt-Sitzungen durchzuführen, um zu einem vergleichbaren Ergebnis zu kommen. Es wird kein Ad-hoc Usability-Test durchgeführt, sondern eine Präsentation (durch die ExpertIn) mit gleichzeitiger bzw. anschließender Diskussion.

**Zweipunkt-Kommunikation möglich**
**Support-Sitzung (HelpDesk)**

Ähnliches gilt für das dritte Szenario, indem die BenutzerIn die Sitzung einleitet, um bei einem konkreten Problem Unterstützung zu erhalten. Sowohl die ExpertIn als auch die EntwicklerIn nehmen an dieser Sitzung teil. Es wäre jedoch möglich, die Vorschläge der ExpertIn und der EntwicklerIn in getrennten Sitzungen einzuholen. Es wird auch hier kein Ad-hoc Usability-Test durchgeführt, sondern eine reine BenutzerInnen-Support-Sitzung.

Es kann von vornherein nicht festgelegt werden, welche Kommunikationsform (Zwei- oder Dreipunkt) eingesetzt wird. Dies ist immer von der konkreten Arbeitsaufgabe abhängig.

- **Bestätigung der notwendigen Durchführungsschritte**
  Die in Kapitel 3.3 erarbeiteten Schritte zur Durchführung eines Ad-hoc Usability-Tests werden durch die Szenarien bestätigt. Obwohl nur das erste Szenario einen „wirklichen" Ad-hoc Usability-Test durchführt, läßt sich auch bei den beiden anderen erkennen, daß zwei Schritte (Kontaktaufnahme und Durchführung) ausreichen, um zu einem adäquaten Ergebnis zu kommen.

**Bestätigung der notwendigen Durchführungsschritte**

- **Untersützung des on-the-fly-Gedankens**
  Anhand der geschilderten Szenarien läßt sich der on-the-fly-Charakter dieser Sitzungen erkennen. Die Semantik des Begriffes on-the-fly definieren wir im Sinne von „beiläufig" und „ohne großen Aufwand".[48] Die Sitzungen können einfach aufgebaut, durchgeführt und eventuell wiederholt werden. Es sind keine aufwendigen Maßnahmen notwendig.

**On-the-fly-Charakter**

## 3.5 Bedarf einer Software zur Unterstützung von Ad-hoc Usability-Tests

**Anforderungen aufsammeln**

Wir werden die einzelnen Anforderungen, die sich aus dem bisherigen Verlauf der Arbeit ergeben haben, in diesem Kapitel „aufsammeln" und überprüfen, inwieweit diese Anforderungen bereits durch gängige Softwaresysteme erfüllt werden bzw. wo und in welchen Bereichen es Bedarf an eigenen Hard- und Softwarelösungen gibt.

**Ist separate Software nötig?**

Wir werden in diesem Kapitel die Frage beantworten, ob mit Hilfe von Standardsoftware und Software, die auf dem freien Markt erhältlich ist (z. B. Freeware, Public Domain), eine ausreichende Unterstützung für Ad-hoc Usability-Tests vorhanden ist oder der Bedarf an einer separaten Software gegeben ist.

### 3.5.1 Datensammlung

**Datenbank für ExpertInnenwissen**

ExpertInnen benötigen bei Ad-hoc-Tests sehr schnellen Zugang zu Richtlinien, Styleguides, Standards und anderen Materialien. Notwendig ist Software, die über Stichwörter und/oder Volltextsuche die Recherche bei bestimmten Problemen erleichtert. Es existiert eine ganze Reihe von Software, die diesen Supportgedanken unterstützt, z. B. verschiedene HelpDesk-Anwendungen.

**HelpDesk**

Die Grundgedanken gängiger HelpDesk-Anwendungen sind (vgl. CYBERCONCEPTS):

- „Die Erreichbarkeit der EDV-Abteilung bei Problemen oder Fragen ist innerhalb der üblichen Arbeitszeiten gewährleistet
- Ein bestimmter Prozentsatz von Anfragen kann unmittelbar ... beantwortet werden.
- Alle nicht unmittelbar zu lösenden Fälle werden an den Second-Level-Support weitergeleitet; dort ist eine Bearbeitung und Lö-

[48] Der on-the-fly-Gedanke wurde bereits in verschiedenen Bereichen eingesetzt (vgl. AOF 1998, MASIE 1996, HOLZMANN 1996).

sung der Fälle innerhalb einer festgelegten Zeitspanne zu gewährleisten.

- Alle Problemmeldungen und Anfragen werden gespeichert; die Lösungen stehen für spätere Nutzung zur Verfügung.
- Die gemeldeten Fälle können ausgewertet werden, um die Ursachen wiederkehrender Probleme zu ermitteln."

Der Markt der HelpDesk-Anwendungen ist in den letzten 3 Jahren gewachsen und wächst weiter. Inzwischen sind auf dem Markt mehr als 250 Werkzeuge von über 100 Herstellern verfügbar (vgl. COMPUTERWOCHE 1996). Einige dieser Werkzeuge werden von einem System benötigt, das Ad-hoc-Tests unterstützt, z. B. eine Datenbasis und Recherchefunktionen.

**Index-Anwendungen**

Weitaus einfachere Möglichkeiten einer Suche in Dokumenten bieten Index-Anwendungen im Web-Umfeld. Die Dokumente werden ungeordnet abgelegt, und ein Tool ermöglicht die Volltextsuche über einen oder mehrere Schlüsselwörter.

### 3.5.2 Remotegedanke

**Räumliche Entfernung**

Softwareentwicklung findet nur selten als „Inhouse-Entwicklung" statt, bei der EntwicklerInnen, ExpertInnen und spätere BenutzerInnen räumlich sehr nahe beieinander sein können. Eine Zusammenkunft der involvierten Personen ist bei räumlicher Entfernung jedoch sehr teuer. Es lohnt sich nicht, bei kleinen Problemen und großen Entfernungen wie zwischen München und Hamburg, zusammenzukommen, um Tests durchzuführen.

**Remote-Charakter**

Das führt uns zum Remote-Charakter von ergonomischen Ad-hoc-Tests. Die beteiligten Personen sind räumlich getrennt und brauchen Möglichkeiten, aus der Entfernung Ad-hoc-Tests durchzuführen. Durch das Internet verfügen wir über Möglichkeiten, diesen Remote-Charakter aufzulösen. Durch die Vernetzung von Rechnern ist eine Beobachtung der BenutzerInnen möglich.

**Tools für entfernte Kommunikation**

Der Remote-Charakter von Ad-hoc-Tests impliziert die Notwendigkeit verschiedener Möglichkeiten, rechnervermittelt zu kommunizieren. Im Bereich von Netzwerken existieren eine Reihe von Tools, die eine derartige Kommunikation unterstützen:

- E-Mail
- Chat
- Audioübertragung
- Videoübertragung
- CSCW-Anwendungen

**Asynchrone vs. synchrone Kommunikation**

In einer Kommunikationsaufbauphase werden im wesentlichen asynchrone Kommunikationsmedien benötigt. In dieser Phase wird beispielsweise das Problem beschrieben. In weiteren Phasen der Kommunikation werden eher synchrone Medien eingesetzt, beispielsweise um einen Termin für den Test zu vereinbaren. Testmethoden wie Lautes Denken benötigen eine ständige Audio- und evtl. auch Videoverbindung.

Im Bereich des TCP/IP-Protokolls existieren Tools, die verschiedene Bandbreiten abdecken (ISDN vs. 34MBit) und verschiedene Anforderungen an das Routing haben (z. B. Multicast). In allen Bereichen werden Tools auch als Freeware angeboten, die für die Zwecke von Ad-hoc-Tests genutzt werden können.

### 3.5.3 Shared Application

**Beobachtung der BenutzerInnen**

Ad-hoc Usability-Tests können nur durchgeführt werden, wenn es gelingt, die Durchführung der Arbeitsaufgabe geeignet zu beobachten. Für diese Zwecke gibt es für herkömmliche Tests Labore, in denen Personen und ihre Arbeit direkt beobachtet werden können. Wenn die Testperson entfernt von den Beobachtenden den Test durchführt, muß Technik eingesetzt werden, um die Beobachtung vorzunehmen. Denkbar wäre die Beobachtung über Video (Videoconferencing Tool).

**Shared application**

*Shared application* macht Anwendungen für entfernte Rechner verfügbar. Die Ein- und Ausgabe eines Programms wird auf den entfernten Rechner übertragen. Eine BenutzerIn kann mit dem Programm so arbeiten, als wenn es auf dem eigenen Rechner gestartet worden wäre.

Möglichkeiten des *application sharing* gibt es für verschiedene Plattformen, z. B. NetMeeting und pcANYWHERE im Windows-Umfeld und SharedX bzw. ShowMe im Unixu-Bereich (vgl. HAMMONTREE, WEILER & NAYAK 1994).

### 3.5.4 On-the-fly-Dokumentation

**Dokumentation während des Tests**

Wie bereits in Kapitel 3.2 beschrieben, ist es bei Ad-hoc-Tests notwendig, die Dokumentation während des Tests mitzuführen. Es gibt kein Tool, das diese Art der Dokumentation wirkungsvoll unterstützt.

**Annotation nachträglich**

Es können Screenshots angefertigt werden, die dann mit gängigen Zeichenprogrammen weiterbearbeitet werden können. Es gibt auch spezielle Tools wie z. B. Wang-Image im Windowsumfeld, mit deren Hilfe eine Annotation von Screenshots durchgeführt werden kann. Diese Art der Annotation würde jedoch keinen on-the-fly-Charakter haben, da die Annotation erst später hinzugefügt wird.

### 3.5.5 Zwischenresümee

**Alle wesentlichen Bereiche sind abgedeckt**

Alle Bereiche, die eine Softwareunterstützung für Ad-hoc-Tests benötigen, können mehr oder weniger gut mit Software, die als Standardsoftware oder Freeware erhältlich ist, abgedeckt werden. Im Gegensatz dazu wird der Bereich der Dokumentation während des Tests nicht von einer derartigen Software abgedeckt. Daraus leitet sich der Bedarf für ein Annotationstool zwingend ab. Die verschiedenen Softwarearten haben eine unterschiedliche Qualität:

**Verschiedene Styleguides**

- Plattform: Einige Softwareprodukte sind für spezielle Betriebssysteme entwickelt worden, andere plattformunabhängig, beispielsweise mit Hilfe von TCL/TK. Das erfordert die Einarbeitung in verschiedene Styleguides und ist nicht benutzerfreundlich.

**Fehlerhafte Software**

- Software, die als Freeware hergestellt ist, hat oft nicht den Anspruch, „fertig zu sein“. Es existieren Fehler, vor allem in der Bedienung, die von den EntwicklerInnen „hingenommen werden“. Im Mittelpunkt einer solchen Software steht oft die Funktionalität und nicht die Bedienung.

**Zu viele Tools**

- Das „Hantieren“ mit fünf bis sechs Softwaretools ist nicht besonders effizient. Mehr als drei offene Fenster auf dem Desktop machen das Arbeiten sehr unübersichtlich, vor allem wenn die Bildschirmauflösung sehr gering ist.

**Bedarf an Software ist vorhanden**

Aus diesen Gründen besteht ein Bedarf an Software zur Unterstützung von Ad-hoc Usability-Tests. Die Software muß folgende Eigenschaften haben:

- **Konsistenz und einfache Benutzbarkeit:**
  Die einzelnen Tools müssen ineinandergreifen und in sich konsistent sein. Neben der Konsistenz müssen die Werkzeuge softwareergonomisch einwandfrei sein. Die Konzentration aller Beteiligten soll auf die Arbeitsaufgabe gelenkt sein und nicht auf die Bedienung der Tools.

- **Zusammenfassung der Benutzungsoberfläche von Tools**

  Unserer Meinung nach ist es sinnvoll, daß einige Tools zusammengefaßt werden, um die Komplexität des Gesamtsoftwareprodukts in bezug auf verschiedene Benutzungsoberflächen zu verringern. Da es in einigen Plattformen Möglichkeiten gibt, nur die Funktionalität von Software ohne die Benutzungsoberfläche zu benutzen (z. B. das ActiveX- oder COM/DCOM-Modell von Windows), kann hiermit Komplexität verringert werden.

# 4 Anforderungen an ein System zur Unterstützung von Ad-hoc Usability-Tests

**Anforderungsdefinition**

Nachdem wir die Notwendigkeit eines Systems zur Unterstützung von Ad-hoc Usability-Tests begründet haben, werden wir nun Anforderungen an dieses System formulieren. In diesem Prozeß wird das Ziel des Systems festgelegt. Die Anforderungsdefinition beschreibt, „was ein System leisten soll, d. h., welche Funktionen das geplante System zur Verfügung stellen soll, aber nicht, wie die einzelnen Funktionen realisiert sind." (POMBERGER & BLASCHEK 1996 S. 43).[49]

**Schritte der Anforderungsermittlung**

Die Anforderungsermittlung haben wir wie folgt durchgeführt:

- Im Mittelpunkt einer Software zur Unterstützung von Ad-hoc Usability-Tests stehen die beteiligten Personen und der Arbeitsgegenstand. Aus den Interessen und Aufgaben der beteiligten Personen ergeben sich Anforderungen an das System.
- Die Personen agieren in einem organisatorischen Umfeld. Sie sind in der Regel weit voneinander entfernt und in ein bestimmtes Arbeitsumfeld eingebunden. Daraus ergeben sich organisatorische Anforderungen an den Kommunikations- und Konferenzaufbau sowie an die Testvorbereitung, -durchführung und -dokumentation.
- Aus spezifischen Interessen und organisatorischen Anforderungen ergeben sich Anforderungen an die Benutzungsoberfläche, die als Vermittlerin zwischen Mensch und Funktionalität fungiert. „Die Oberfläche muß deshalb eine effektive Übersetzung in beide Richtungen realisieren, um einen erfolgreichen Dialog zu ermöglichen." (DIX et al. 1995 S. 118).

**Allgemeine vs. konkrete Anforderungen**

Zu einer Anforderungsermittlung gehören sowohl allgemeine Anforderungen, z. B. „Sollen Fehler in geeigneter Weise angezeigt und Lösungen angeboten werden", sowie konkrete Anforderungen, wie „Die Bildschirmauflösung muß 1024 x 768 Bildpunkte betragen".

**Tabellarische Auflistung**

## 4.1 Allgemeine Anforderungen

Wir gliedern die Anforderungen analog der Durchführung der Konferenz in allgemeine Anforderungen, Voraussetzungen der Kontaktaufnahme, Kontaktaufnahme und Konferenzaufbau, Testdurchführung, Dokumentation, Statusanzeigen und Anforderungen an die Benutzungsoberfläche.

Wir unterscheiden zwischen einem Basissystem und Systemerweiterungen, ohne die konkrete Ausprägung des Systems bereits zu diesem Zeitpunkt zu spezifizieren. Das Basissystem enthält alle Anforderungen, die für eine Unterstützung von Ad-hoc Usability-Tests notwendig sind. Die Systemerweiterungen beschreiben Anforderungen, die sinnvoll wären („*nice to have*"), aber nicht substantiell sind. Einige Anforderungen gelten nicht für alle Personengruppen

[49] POMBERGER & BLASCHEK nennen den Prozeß der Anforderungsermittlung „Systemspezifikation" (1996).

oder gelten für einige Personengruppen besonders. Dies ist bei den einzelnen Anforderungen gesondert vermerkt, ansonsten können sie auf alle Personengruppen bezogen werden.

**1. Zielplattform: Windows9x** **Basissystem**

Die Zielplattform des Systems ist Windows95/98. Dies begründet sich mit der marktbeherrschenden Stellung im Bereich der Betriebssysteme Windows9x, Windows NT, Windows CE (vgl. COMPUTERNEWS 1998) sowie der Office-Applikationen[50] (vgl. BAGER 1997). Aufgrund dieser Tatsache werden auch die meisten Anwendungen für Windows entwikkelt. Auf Besonderheiten von Windows NT ist auf der EntwickerIn- und der ExpertInseite Rücksicht zu nehmen.

**2. Notwendigkeit einer Netzwerkverbindung** **Basissystem**

Da die Teilnehmenden der Konferenz räumlich getrennt agieren, muß zwischen ihnen ein Medium eingesetzt werden, daß eine Zusammenarbeit ermöglicht und die räumliche Entfernung überbrückt. Aus diesem Grund wird ein Netzwerk benötigt, das diese Funktion übernimmt.

**3. Eigenschaften der Netzwerkverbindung** **Basissystem**

Damit eine Konferenz durchgeführt werden kann, hat die eingesetzte Netzwerktechnik folgenden Eigenschaften zu genügen:

a) Der Zugang zum Netzwerk ist jederzeit mit vertretbarem Aufwand möglich.

b) Es sind mehrere gleichzeitige Verbindungen zu verschiedenen Adressaten möglich.

c) Das Netzwerk bildet gleichzeitig verschiedene Datenstromrichtungen (z. B. Audio-Streams) ab.

d) Das Netzwerk stellt eine Mindestbandbreite für die Übertragung zur Verfügung.

e) Die Daten (z. B. Audio- und Videodaten) werden in Echtzeit ohne Qualitätsverluste übertragen.

**4. Audioübertragung** **Basissystem**

Das System muß die Äußerungen der BenutzerInnen während der Aufgabendurchführung an die anderen TeilnehmerInnen der Konferenz synchron übermitteln. So kann einerseits bei räumlicher Trennung der TeilnehmerInnen das „Vereinfachte Laute Denken“ des Ad-Hoc Usability-Tests eingesetzt werden. Andererseits wird die „normale“ Konversation zwischen den Konferenz-TeilnehmerInnen ermöglicht, die sich in der Phase der Kontaktaufnahme näher kennenlernen und miteinander kommunizieren können.

[50] MS-Office 9x.

**5. Videoübertragung** **Basissystem**

Das System muß eine Video-Erfassung und Übertragung unterstützen, da der Ad-hoc Usability-Test über die reine Aufgabenbeobachtung auch eine BenutzerInnen- und Umfeld-Beobachtung vorsieht. Durch die Übertragung der Videobilder, die bei der BenutzerIn aufzeichnet werden, können die ExpertIn bzw. die EntwicklerIn die Arbeit und das Verhalten analysieren.

Die Video-Übertragung kann auch benutzt werden, um eine Konversation zwischen den Teilnehmenden, über den verbalen Aspekt der Audio-Übertragung hinaus zu ermöglichen. Für diesen Zweck muß darauf geachtet werden, daß die Audio- und Video-Daten synchronisiert werden, da sie nur im direkten Zusammenspiel brauchbar sind.

**6. Datenübertragung** **Basissystem**

Es ist erforderlich, daß Daten an bestimmte oder an alle Teilnehmenden der Konferenz korrekt[51] übertragen werden. Dieser Übertragungs-Mechanismus ist durch das System bereitzustellen. Beispielsweise werden für die Durchführung des Tests Daten (z. B. Dokumente) von allen Teilnehmenden benötigt, die aber nur bei einer Person vorliegen, oder es fallen während des Tests Daten (z. B. Video-Mitschnitte) an, die ebenfalls verteilt werden müssen.

**7. Rechnerausstattung** **Basissystem**

Aufgrund der hohen Komplexität der Konferenz und der damit verbundenen Anzahl von (Status-) Informationen soll das eingesetzte Rechner-System zumindest bei der ExpertIn zwei Bildschirme unterstützen und einsetzen, so daß alle notwendigen Anzeigen verteilt und überwacht werden können.

Zudem müssen die Audioaufnahme und -wiedergabe-komponenten innerhalb des Rechners (z. B. Soundkarte) voll duplex-fähig (gleichzeitige Aufnahme und Wiedergabe) und mindestens Stereoqualität (zwei getrennte Kanäle) bieten. Damit ist gewährleistet, gleichzeitig „zu hören und zu sprechen“ bzw. verschiedene Personen gleichzeitig zu hören. Für die externe Sprachaufnahme bzw. Wiedergabe müssen entsprechende technische Ausstattungen (z. B. Headset) vorhanden sein.

Die installierte visuelle Wiedergabekomponente (z. B. Grafikkarte) muß in der Lage sein, mindestens 65.536 Farben darzustellen.

Weiterhin wird eine interne Videokarte benötigt, die externe Signale einer Videokamera aufnimmt und verarbeitet. Damit bei der BenutzerIn die Arbeitsumgebung nicht unnötig verändert werden muß, ist die Videokamera in die Rechnerausstattung zu integrieren (z. B. im Monitor einzubauen).

[51] Korrekt im Sinne von sicher und zuverlässig.

**8. Angepaßte Computerleistungsfähigkeit** **Basissystem**

Durch die hohe Komplexität der gesamten Konferenz- und Testsituation wird eine besonders hohe Leistungsfähigkeit der eingesetzten Computer vorausgesetzt. Nur so ist es möglich, den Test effektiv und effizient durchzuführen. Das Maß der Leistung der Rechner hängt dabei unmittelbar vom gewünschten Verhalten des Systems (z. B. Antwortzeiten) ab.

**9. Bildschirmauflösung** **Basissystem**

Aus ergonomischen Gründen werden als Mindestanforderung an Bildschirme 19“ Monitore, mit einer Bildschirmauflösung von 1024 x 768 Bildschirmpunkten gefordert.

Die ExpertInnen benötigen entweder eine höhere Auflösung oder eine Zwei-Bildschirm-Lösung (vgl. Punkt 7), weil sie deutlich mehr Informationen als die EntwicklerInnen oder die BenutzerInnen benötigen.

**10. Reduzierung der Technik auf das Notwendige** **Basissystem**

Die eingesetzte Technik muß auf die für die Durchführung notwendige Größe zugeschnitten und beschränkt bleiben Die erforderliche Ausstattung soll überschaubar und die notwendigen Kosten für die Durchführung eines Ad-hoc Usability-Tests möglichst gering gehalten werden.

**11. Durchführbarkeitsprüfung auf Systemebene** **Erweiterung**

Bevor das System die eigentliche Konferenz aufbaut, soll eine Überprüfung der Durchführbarkeit stattfinden. Aufgrund der Anforderungen an ein System zur Unterstützung von Ad-hoc Usability-Tests muß das System selbsttätig prüfen, ob alle Voraussetzungen (z. B. Netzwerkverbindungen, Geräteverfügbarkeit) erfüllt sind, die Verbindungen zu erstellen.

**12. Qualitative Anforderungen an Datenströme** **Basissystem**

Die qualitativen Anforderungen an die verschiedenen Datenströme (Audio, Video, verteilte Applikationsdarstellung) sind:

- Audioübertragung mindestens 8 Bit Mono, weil das Verhältnis von Datenaufkommen und Verständlichkeit hier adäquat ist.
- Videoübertragung mindestens fünf Bilder/Sekunde. Dies erlaubt eine Erkennung von Bewegungsabläufen (z. B. Mimik, Gestik).
- Übertragung der verteilten Applikationsdarstellung mindestens zwei Bilder/Sekunde, um den Arbeitsablauf verfolgen zu können.

**13. Datenschutz** **Basissystem**

Das gesamte System muß den Datenschutzrichtlinien entsprechen, beispielsweise muß der Schutz vor Leistungs- und Verhaltenskontrolle garantiert sein.

**14. Transparenz** **Erweiterung**

Es soll zu jedem Zeitpunkt erkennbar sein, von wem welche Aktionen bzw. Daten mitgeschnitten und aufgezeichnet werden.

**15. Daten müssen einsehbar sein** **Basissystem**

Alle Beteiligten haben jederzeit die Möglichkeit, die Daten oder die Handlungen, die über sie gespeichert sind, einzusehen. Dazu ist ein geeigneter Zugriff auf die Daten zu ermöglichen.

**16. Sicherheitsaspekte** **Erweiterung**

Während des Tests werden vertrauliche und sensible Daten (z. B. der Prüfgegenstand) zwischen den Teilnehmenden übertragen. Diese dürfen nicht von unbefugten Personen eingesehen werden (Datenmißbrauch).

Darüber hinaus basiert die gemeinsame Arbeit auf gegenseitigem Vertrauen, so daß eine unbefugte Nutzung des Systems ausgeschlossen werden muß (Systemmißbrauch).

**17. Medienwechsel** **Basissystem**

Aus didaktischer Sicht ist es sinnvoll, bei der Erläuterung von Testergebnissen auf verschiedene Darstellungsformen zurückzugreifen (z. B. skriptural, verbal). Zu diesem Zweck sind verschiedene Medien für die Verdeutlichung von Testergebnissen zur Verfügung zu stellen.

Diese Anforderung gilt vor allem für die ExpertIn und EntwicklerIn, die Testergebnisse diskutieren.

**18. Notizmöglichkeit** **Erweiterung**

Während des Tests soll es möglich sein, allgemeine Notizen (skriptural, verbal) zum Testverlauf zu machen. Bedingt durch die Komplexität des Tests und dem hohen Grad von parallelen Aktivitäten (z. B. visuelle und akustische Beobachtung, Steuerung des Tests) der ExpertInnen, stellt das System für diese Zwecke einen einfachen und leicht bedienbaren Mechanismus bereit (z. B. Aktivierung und Steuerung per verbaler „Kommandosprache“).

Diese Anforderung gilt im wesentlichen für die ExpertIn.

**19. Schnittstelle für die Heuristische Evaluation** **Erweiterung**

Das System soll eine Schnittstelle zur Bereitstellung von Heuristiken bzw. zu einem System anbieten, mit dem eine Heuristische Evaluation durchgeführt werden kann. Dies kann z. B. ein Fragebogen in Form einer Excel-Tabelle sein.

Diese Anforderung gilt nur für die ExpertIn.

**20. Skalierbarkeit und Mindestanforderungen** **Basissystem**

Die eingesetzte Technik muß in ihrem Einsatz skalierbar sein, d. h., sie ist gemäß dem jeweiligen Anwendungszweck und den technischen Rahmen-

bedingungen anpaßbar. So entsteht ein Intervall zwischen notwendigen technischen Mindestanforderungen und verfügbaren technischen Ressourcen.

Die Skalierung der Technik muß sich in diesem Intervall bewegen, um eine prinzipielle Testdurchführung zu ermöglichen. Dies gilt insbesondere für die Übertragung von Daten (z. B. Audio) in Netzwerken. Hierfür wird auf der einen Seite eine bestimmte Bandbreite zur Erreichung einer Mindestqualitität benötigt, während auf der anderen Seite die technischen Ressourcen effizient ausgenutzt werden müssen, da sie Kosten verursachen. Bei der Wahl der Ressourcen ist darauf zu achten, daß sie allgemein verfügbar sind (z. B. ISDN).

## 4.2 Voraussetzungen der Kontaktaufnahme

**21. Konferenzkonfigurationen erstellen** **Basissystem**

Bereits bei Planung eines Software-Projekts wird festgelegt, welche Personen die Rolle der ExpertIn, des Entwickelnden und der BenutzerIn übernehmen. Die persönlichen Daten dieser Personen (z. B. Rolle, IP-Adresse, Telefonnummer) sind in einer geeigneten Form erfaßbar zu machen und persistent zu halten. Wir nennen diese Daten im folgenden „Konferenzkonfiguration".

Die Konferenzkonfiguration ist vorzugsweise von der ExpertIn zu erstellen und persistent zu halten.

**22. Verwaltung von Konferenzkonfigurationen** **Basissystem**

Es muß möglich sein, mehrere Konferenzkonfigurationen anzulegen, d. h., das System soll diverse Konferenzen verwalten können.

## 4.3 Kontaktaufnahme und Konferenzaufbau

**23. Einfache und schnelle Kontaktaufnahme** **Basissystem**

Der Kommunikations- und Konferenzaufbau muß einfach und schnell sein.

Besonders bei den BenutzerInnen müssen Möglichkeiten geschaffen werden, ohne Aufwand sowohl den Kommunikations- als auch den Konferenzaufbau zu vollziehen bzw. nachvollziehen zu können.

**24. Verteilte Kooperation** **Erweiterung**

Über die Kommunikationsunterstützung mittels Audio- und Videoübertragung soll es möglich sein, miteinander an gemeinsamen Arbeitsgegenständen zu arbeiten. Das ist mit der verteilten Applikationsdarstellung möglich, indem derartige Applikationen (z. B. Textverarbeitungen) bei allen Teilnehmenden dargestellt werden. Da dies die Zusammenarbeit nur ermöglicht, aber nicht ausreichend unterstützt, muß das System explizite und geeignete Möglichkeiten hierfür anbieten.

Diese Anforderung gilt besonders für die ExpertIn und die EntwicklerIn.

**25. Terminfindung** **Basissystem**

Die verschiedenen Personen müssen sich auf einen gemeinsamen Termin einigen. Dafür sind synchrone (z. B. Telefon) und asynchrone Mechanismen (z. B. E-Mail) erforderlich. Das Telefon kann in die Konferenzsteuerung integriert sein.

**26. Konferenzstruktur** **Basissystem**

Die eigentliche Konferenz wird strukturiert, indem die spezifischen Rollen allen mitgeteilt werden. Außerdem wird diese Struktur bei den Teilnehmenden der Konferenz integriert, damit das System auf diese Daten zurückgreifen kann. Das System unterstützt diese Erstellung, Verteilung und Integration.

Die Strukturierung wird von der ExpertIn vorgenommen.

**27. Konferenzeinladung** **Basissystem**

Die Aktion der Einladung aller Teilnehmenden zur Konferenz ist durch das System zu ermöglichen. Hierzu werden angepaßte Statusmeldungen abgesetzt und angezeigt. Allen Teilnehmenden der Konferenz werden Informationen durch das System bereitgestellt, z. B. wer aktuell an der Konferenz teilnimmt oder wer eine Teilnahme abgelehnt hat.

**28. Begrüßungsphase** **Basissystem**

Das System muß es ermöglichen, daß sich die Teilnehmenden vor dem eigentlichen Testbeginn begrüßen und unterhalten können. Dies verstärkt die für den Test notwendige Vertrauensbasis und wirkt sich somit direkt auf die zu erarbeitenden Ergebnisse aus.

**29. Konfiguration des Tests** **Erweiterung**

Der Testgegenstand und alle notwendigen Utensilien werden verteilt. Die Erstellung einer Kombination und Konfiguration dieses Bestandteils des Tests ist durch das System zu unterstützten. So soll beispielsweise die Applikation, die als Testgegenstand untersucht wird, zu jedem Zeitpunkt komplett einsehbar sein und darf nicht von anderen Darstellungen überdeckt sein. Außerdem wird diese Konfiguration von der ExpertIn mittels des Systems an alle Beteiligten verteilt.

**30. Verteilung der Arbeitsaufgabe** **Basissystem**

Die Arbeitsaufgabe muß bekanntgegeben bzw. diskutiert werden. Das Testziel muß sich erkennen lassen können. Zusätzlich werden noch die Testgegebenheiten (z. B. Verwendung der erfaßten Daten) benannt und erörtert. Das System unterstützt diese Diskussion in geeigneter Weise.

## 4.4 Testdurchführung

**31. Aus- und Wiedereinstieg in die Testphase** **Basissystem**

Während der Testphase können einzelne Teilnehmende die Konferenz verlassen oder verspätet beitreten.

Die EntwicklerInnen beispielsweise interessieren sich unter Umständen nur für Testergebnisse bzw. für bestimmte Teilsequenzen des Tests.

**32. Test darf Arbeitsablauf nur wenig stören** **Basissystem**

Die Tests dürfen den normalen Arbeitsablauf der BenutzerInnen nur wenig stören. Die Arbeit soll nur für die Durchführung der Arbeitsaufgabe unterbrochen werden.

**33. Systemperformance** **Basissystem**

Das Testsystem darf die Durchführung des Tests nicht behindern (z. B. durch eine Verschlechterung der Systemperformance), weil dies zur Verzerrung der Testergebnisse führen würde.

**34. Testbeginn und -ende** **Erweiterung**

Für alle soll deutlich sein, wann der Test und damit die Aufzeichnung beginnt und wann er endet. Alle können den Test unterbrechen (z. B. bei einem Telefongespräch der BenutzerIn).

**35. Verteilte Applikationsdarstellung** **Basissystem**

Der Arbeitsgegenstand des Tests ist ein Prototyp (Prüfgegenstand) bestimmter Güte, der durch den Entwickelnden bereit gestellt wird. Das System muß eine Möglichkeit bieten, die Arbeit der BenutzerInnen an die anderen Teilnehmenden zu übertragen. Die anderen Teilnehmenden können somit die Arbeitsdurchführung der BenutzerInnen aus der Distanz verfolgen und analysieren, um die Aufgabenbeobachtung des Ad-hoc Usability-Tests aufzuführen.

Es gibt allerdings Situationen, in denen es sinnvoll sein kann, daß andere Personen steuernd und kontrollierend auf die Applikation zugreifen. Dies ist beispielsweise der Fall, wenn eine Arbeitssequenz nicht zu Ende geführt werden kann und das System in einen bestimmten Zustand für die nächste Sequenz gebracht werden muß.

**36. Aktivierung des Testgegenstands** **Basissystem**

Das System aktiviert vor dem tatsächlichen Testbeginn bei allen Teilnehmenden alle notwendigen Utensilien des Tests (z. B. Applikationen, Dokumente), damit der Test begonnen werden kann. Dies geschieht zeitgleich, damit alle Teilnehmenden den Test unter gleichen Bedingungen beginnen können.

**37. Anzeige des Testsverlaufs** **Erweiterung**

Das System soll anhand der Arbeitsaufgabe einen Testverlauf strukturieren und während des Tests darstellen, welche Teile des Tests bereits abgearbeitet wurden (z. B. mittels Zeitinformationen oder Meilensteine). Darüber hinaus soll es möglich sein, in jeder Testphase die jeweiligen Ergebnisse anzuzeigen bzw. abzurufen.

Die Anzeige des Verlaufs ist im wesentlichen für die ExpertIn notwendig.

**38. Datenzustand und deren Verfügbarkeit** **Erweiterung**

Alle Daten, die während des Tests in verschiedenen Streams (z. B. Audio-, Video- oder Applikationsdaten) anfallen, sollen bereits während des Tests ohne weitere Aktionen persistent und verfügbar sein.

Dies gilt insbesondere für die ExpertIn.

**39. Bestimmung des Ziels der verbalen Äußerungen** **Erweiterung**

Das System soll eine Auswahl der EmpfängerInnen (Makeln, Whisper-Modus) ermöglichen. Ergänzend hierzu bietet das System eine einfache Möglichkeit an, die Übertragung von Daten kurzzeitig (z. B. aufgrund einer externen Störung) zu unterbrechen.

**40. Wahl des Empfangs von verbalen Äußerungen** **Basissystem**

Das System bietet den Teilnehmenden eine Einstellmöglichkeit an, mit der sie wählen können, ob sie die verbalen Äußerungen hören bzw. empfangen möchten. Zusätzlich ist es möglich, gleichzeitig Audio-Daten von mehreren Quellen zu empfangen, ohne daß sie sich vermischen.

**41. Einstellungen der Audio-Empfangsqualität** **Basissystem**

Das System muß eine Möglichkeit bieten, die Qualität der Übertragung der verbalen Äußerungen einzustellen. Hierzu zählen beispielsweise die Lautstärke der Aufzeichnung bzw. der Wiedergabe.

**42. Bestimmung des Ziels der Beobachtungsdaten** **Erweiterung**

Das System soll eine Möglichkeit bieten auszuwählen, an wen die Daten temporär bzw. grundsätzlich gesendet werden (Makeln). Ergänzend hierzu soll das System eine einfache Möglichkeit anbieten, die Übertragung der Daten kurzzeitig (z. B. aufgrund einer externen Störung) zu unterbrechen.

**43. Wahl des Empfangs der Videodaten** **Erweiterung**

Das System soll den Teilnehmenden die Wahl ermöglichen, ob sie die Bearbeitung der Testapplikation (Aufgabenbeobachtung) bzw. die Beobachtung der BenutzerIn (Personenbeobachtung) empfangen möchten.

**44. Einstellungen des Empfangs der Videodaten** **Basissystem**

Das System muß eine Möglichkeit bieten, die Qualität (z. B. Helligkeit, Größe) der Übertragung der eigenen, zu übertragenen Daten einzustellen.

**45. Phasen der Prüfaufgabe** **Erweiterung**

Zur Unterstützung der Dokumentation des Tests soll das System einzelne Phasen der Bearbeitung der Prüfaufgabe unterscheiden können (z. B. erster Dialog, zweiter Dialog, ...), die vor Testbeginn im Rahmen der Arbeitsaufgabe festgelegt werden. Anhand dieser Unterscheidungen kann die Dokumentation gegliedert und erstellt werden.

## 4.5 Dokumentation

**46. Allgemeine und konkrete Hinweise** **Basissystem**

Die Dokumentation des Tests kann je nach formuliertem Ziel aus allgemeinen Gestaltungshinweisen bestehen, muß aber auch konkrete Anleitung geben, wie die Qualität des Produkts verbessert werden kann.

Dieser Aspekt gilt für die ExpertIn.

**47. Aufbereitete Dokumentation und Originalmaterial** **Erweiterung**

Die Dokumentation besteht sowohl aus aufbereiteter Dokumentation (wie z. B. Annotationen) als auch aus Originalmaterial (z. B. Mitschnitt von Audiodaten).

**48. Referenzierbarkeit der Dokumente** **Erweiterung**

Es soll möglich sein, von bestimmten Aspekten in der Dokumentation (z. B. Screenshots) auf das Originalmaterial zu schließen.

**49. Audio-Wiedergabe und -Bearbeitung** **Erweiterung**

Über reine Übertragung und damit verbundene Wiedergabe der übertragenen Daten soll das System auch die Bearbeitung und Wiedergabe von aufgezeichneten (z. B. der mitgeschnittenen) Übertragungen ermöglichen. Diese Audio-Daten können beispielsweise für eine Analyse und die spätere Dokumentation aufbereitet, zusammengestellt und genutzt werden.

**50. Parallele Erzeugung der Dokumentation** **Basissystem**

Das System muß die Erstellung der Dokumentation in der Form unterstützen, daß bei der Anfertigung keine systematische oder zeitliche Trennung vom eigentlichen Test notwendig wird. Die Mechanismen der Dokumentation werden vollständig in den Ablauf des Tests integriert, um eine parallele Erstellung zu ermöglichen.

Die parallele Erzeugung der Dokumentation findet durch die ExpertIn statt.

**51. Darstellung aller Daten in ihrem Kontext** **Basissystem**

Es ist zu jedem Zeitpunkt möglich, alle verfügbaren Daten in ihrem jeweiligen Kontext darzustellen (z. B. hierarchische Ansicht) bzw. ihre Inhalte anzuzeigen. Hierdurch wird unterstützt, beliebige Verknüpfungen zwischen diesen Daten zu erstellen und in die Dokumentation zu integrieren.

**52. Bearbeitungsmöglichkeit der verfügbaren Daten Erweiterung**

Das System soll es ermöglichen, beliebige Kombinationen von Daten anzufertigen und diese mit zusätzlichen Kommentaren in die Dokumentation zu übernehmen.

## 4.6 Statusanzeigen

Das System stellt verschiedene Informationen über den aktuellen Systemstatus bereit, die im wesentlichen von der ExpertIn benötigt werden.

- Informationen über die Konferenz (insbesondere der aktuelle Name)
  Da das System die Erstellung und Vorbereitung mehrerer Konferenzen ermöglicht, sind Informationen (z. B. Konferenzname) über die aktuelle Konferenz anzuzeigen.
- Statusanzeige der verteilten Applikation
  Das System muß nicht nur die verteilten Applikationen darstellen (siehe oben), sondern darüber hinaus auch Informationen (z. B. aktuelle BenutzerIn) und Zustände der Applikationen (z. B. Präsentations- oder Mitbenutzungsstatus) für alle Teilnehmenden zur Verfügung stellen.
- Statusanzeige des Audio- bzw. Videostatus
  Der aktuelle Status von Audio (z. B. visuelle Darstellung des aktuell Sprechenden) und Video (z. B. Name des Dargestellten) muß vom System angezeigt werden.
- Statusinformationen über die Teilnehmenden
  Neben der Statusanzeige des Audio-/Videostatus gewährleistet das System auch Informationen über den Status der Teilnehmenden der Konferenz. Hierzu zählt beispielsweise, wer aktuell an der Konferenz teilnimmt oder wer die Konferenz verläßt bzw. temporär unterbricht. Diese Informationen bzw. Statusänderungen werden, sowohl visuell als auch akustisch angezeigt und signalisiert.
- Zuordnung der Äußerungen zur Senderquelle
  Das System zeigt grundsätzlich an, wer den dargestellten Bildern „zugeordnet", also die Quelle der Daten ist.

## 4.7 Anforderungen an die Benutzungsoberfläche

**BenutzerInnen und Aufgaben stehen im Mittelpunkt**

„Die Benutzerschnittstellen repräsentieren eine benutzerorientierte Abstraktion der Funktionalität eines Systems. Ihre Gestaltung beeinflußt maßgeblich die Akzeptanz eines Softwareprodukts." (POMBERGER & BLASCHEK 1996 S. 44). Aus diesem Grunde muß auch die Oberfläche eines Ad-hoc Test-Systems den in Kapitel 2 erarbeiteten allgemeinen Ansprüchen genügen.

### 4.7.1 Oberflächengestaltung

**Styleguide der Plattform anpassen**

Die Oberfläche muß dem jeweiligem Styleguide der Entwicklungsplattform entsprechen. Im wesentlichen sind Funktionen und Aufgaben so abzubilden, daß sie mit den Standardkomponenten der jeweiligen Plattform gestaltet wer-

den können. Bevor neue Komponenten gestaltet werden, müssen folgende Fragen beantwortet werden (vgl. MICROSOFT 1995. 17):

- „Wird die benötigte Funktionalität definitiv nicht bereits in anderer Form von der Oberfläche geboten - eventuell durch eine Kombination mehrerer Komponenten?
- Ist das Verhalten der neuen Komponente für den Benutzer vorhersagbar - oder steht es etwa im Widerspruch zu Standardelementen?
- Fügt sich die neue Komponente optisch in die Oberfläche ein?
- Hebt sich die Komponente optisch genügend von existierenden Komponenten ab, um Mißverständnisse von seiten des Benutzers zu vermeiden?“

### 4.7.2 Integration der einzelnen Funktionen

**Datenorientiertes Modell**

Bei der Umsetzung des Systems sollen nicht nur aus Kostengründen, sondern insbesondere im Hinblick auf universelle Verwendbarkeit Funktionalitäten von Standardprogrammen und frei erhältlichen Programmen in das Gesamtsystem integriert werden. Dabei soll möglichst nur die Funktionalität, nicht aber die Benutzungsoberfläche der verschiedenen Softwareprodukte eingebettet werden, sofern diese Oberflächen nicht styleguidekonform gestaltet sind. Im Mittelpunkt steht ein datenorientiertes Modell: „Anwendungen übernehmen immer stärker die Rolle eines Uhrwerks im Hintergrund einer Umgebung, in der sich die Benutzer auf die Manipulation ihrer Daten - im Gegensatz zur Arbeit mit bestimmten Programmen - konzentrieren.“ (MICROSOFT 1995 S. 15)

**Einbettung von Werkzeugen**

Ein wesentlicher Teil eines Systems zur Unterstützung von Ad-hoc Usability-Tests ist die Konferenzsteuerung. Alle anderen Werkzeuge und Funktionalitäten werden über die Konferenzsteuerung aktiviert und sind in ihr eingebettet. Die Benutzungsoberfläche stellt eine Schnittstelle zu anderen Programmen zur Verfügung. So können auch nachträglich Werkzeuge in das System eingebettet werden.

**Grad der Integration ist abhängig vom informatischen Fachwissen**

Der Grad der Integration ist vom jeweiligen Personenkreis abhängig:

- Bei der ExpertIn und EntwicklerIn ist dieser Grad nicht notwendigerweise hoch, da sie über explizites informatisches Fachwissen verfügen, das die Unterschiede im Erscheinungsbild und in der Benutzung kompensieren kann.
- Bei der BenutzerIn hingegen muß dieser Integrationsgrad extrem hoch sein, da hier in der Regel keine umfassenden informatischen Fachkenntnisse vorausgesetzt werden können, die diese Unterschiede überwinden können.

### 4.7.3 Konkrete Anforderungen

**Rollenabhängige Anforderungen**

Die konkreten Anforderungen sind rollenabhängig, da die unterschiedlichen Personenkreise differenzierte Zielsetzungen haben. Diese Zielsetzungen manifestieren sich einerseits durch die angebotene Funktionalität des System, wäh-

rend andererseits die Benutzungsoberfläche diese Zielsetzungen unterstützen und umsetzen soll.

- Auf der ExpertIn-Seite muß eine hohe Dichte der Funktionalität in der Benutzungsoberfläche erreicht werden, um die hohe Komplexität der Testdurchführung zu ermöglichen bzw. unterstützen.
- Es bedarf einer extrem hohen Selbstbeschreibungsfähigkeit der Oberfläche auf BenutzerIn-Seite, da die Benutzung dieses Systems normalerweise nur sporadisch erfolgt. Um trotzdem eine effektive Testdurchführung zu gewährleisten, müssen die BenutzerInnen das System leicht benutzen können.
- Die Aufgabenangemessenheit muß auf ExpertInnen- und EntwicklerInnen-Seite verstärkt im Mittelpunkt stehen, denn die Aspekte des Ad-hoc-Tests sollen eine effektive und effiziente Durchführung des Tests sicherstellen
- Da alle drei Parteien verteilt und getrennt agieren, muß das System allgemein und die Benutzungsschnittstelle insbesondere eine Fehlerrobustheit gewährleisten, um die zügige und ungestörte Durchführung des Tests zu unterstützen.
- Die Benutzungsschnittstelle muß dem Prinzip der Individualisierbarkeit genügen, da aufgrund der unterschiedlichen Rollen und Zielsetzungen individuelle Einstellmöglichkeiten die Effizienz des Systems auf allen Seiten unterstützen.
- Die Oberfläche muß über die Selbstbeschreibungsfähigkeit hinaus einen hohen Grad der Erlernbarkeit gewährleisten. Dies unterstützt den Einsatz des Systems besonders bei längeren Projekten, die dieses System mehrmals einsetzen.

# 5 Umsetzung der Anforderungen- ErgoNet

**Ebenen unseres Softwareentwurfs**

In diesem Kapitel geht es um die Umsetzung der Anforderungen in ein konkretes System. Wir unterscheiden in unserem Softwareentwurf vier Ebenen (vgl. POMBERGER & BLASCHEK 1996 S. 57). Diese Ebenen haben wir für die Gliederung der Darstellungen unseres Systems eingesetzt:

1. **Das Steuermodul**
   Dieses Modul ist für Koordinationsaufgaben, für Initialisierungen und Abschlußarbeiten zuständig.
2. **Problemorientierte Module**
   Mit diesen Modulen werden Teilaufgaben des Systems bearbeitet. Diese Ebene dient der eigentlichen Problemlösung. Solche problemorientierten Module sind z. B. Audio, Video, *shared application* und die Konferenzsteuerung.
3. **Hilfsmodule**
   Auf dieser Ebene befinden sich Listen, Tabellen und andere interne Datenstrukturen zur Verwaltung.
4. **Hardware**
   Die unterste Schicht bildet die Module zur Kommunikation mit der Hardware und dem Betriebssystem.

**Prototyporientierte Systemspezifikation**

Die Umsetzung des Systems haben wir mit Hilfe der prototyporientierten Systemspezifikation durchgeführt[52]. Wir haben in diesem Kapitel mehrere Iterationsschritte zusammengefaßt:

1. Schritt: ActiveX Steuerung von Netmeeting. Vordringlich war für uns die Ansteuerung von MS-Netmeeting.
2. Schritt: COM-Steuerung von Netmeeting.
3. Schritt: Konferenzsteuerung und Annotation zur Präsentation auf der CeBIT'98.
4. Schritt: Überarbeitung der Benutzungsschnittstelle für die Präsentation beim Tag der offenen Tür (TZI).

Das System heißt in Anlehnung an die Überlegungen beim ISI/TZI[53] ErgoNet. Mit diesem Namen wird sowohl die Intension (Software-Ergonomie) als auch der Netzcharakter des Systems betont.

## 5.1 Technische Umsetzung

Nachfolgend erläutern und begründen wir unsere Entscheidungen für den Einsatz bestimmter Techniken bei der Umsetzung des Systems zur Unterstützung von Ad-hoc Usability-Tests. Darüber hinaus beschreiben wir die jeweiligen Techniken inhaltlich, ohne auf die konkreten technischen Details einzugehen.

[52] Zur näheren Erläuterung dieses Verfahrens vgl. Kapitel 6.2.1.

[53] Institut für Software-Ergonomie und Informationsmanagement beim Technologie-Zentrum Informatik.

### Betriebssystemwahl

Das zugrundeliegende Betriebssystem ist Microsoft Windows9x.

**Bedürfnis für Klärung bei ergonomischen Sachfragen**

Mit dem Betriebssystem Windows, seiner graphischen Oberfläche und den vielfältigen Entwicklungen dieses Systems steht ein potentielles Anwendungsgebiet zur Verfügung, auf dem ein Bedürfnis zur Klärung von ergonomischen Sachverhalten besteht.

Da davon auszugehen ist, daß die BenutzerInnen ebenfalls über Kenntnisse des Betriebsystems Windows und seiner Benutzung verfügen, muß das System selbst auch auf dieser Plattform umgesetzt werden, damit Kenntnisse anderer Betriebssysteme überflüssig sind.

### Microsoft Netmeeting

**Netmeeting-Einsatz, um verteilte Applikationsdarstellung zu ermöglichen**

Es besteht die Anforderung an das System, eine verteilte Applikationsdarstellung anzubieten, um die konkrete Aufgabenbeobachtung durch die EntwicklerInnen und ExpertInnen durchzuführen. Das Produkt Netmeeting[54] von Microsoft stellt diese Funktionalität bereit und wird kostenlos von Microsoft zur Verfügung gestellt bzw. ist in Windows bereit integriert.

Netmeeting besteht aus mehreren Komponenten, die sich folgendermaßen zusammensetzen:

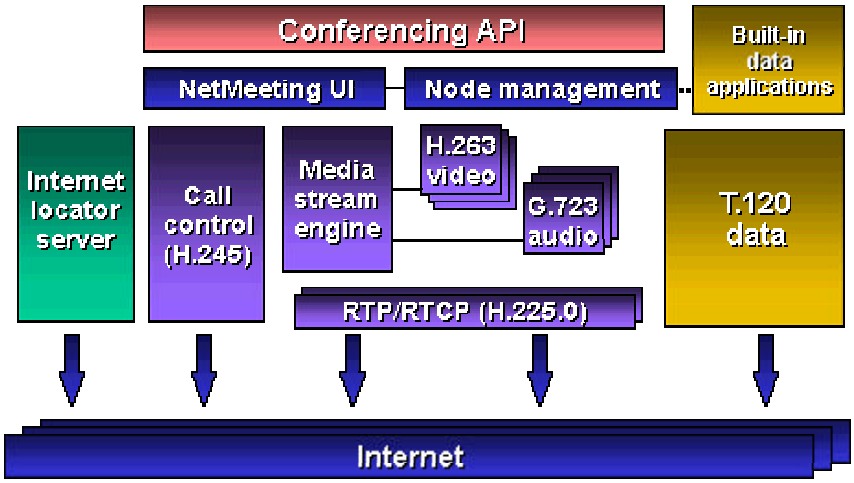



Abbildung 8: Netmeeting-Architektur (NETMEETING SDK 1997)

In unserem System werden dabei allerdings aus verschiedenen Gründen, die nachfolgend erläutert sind, nur die folgenden aktiviert und eingesetzt:

[54] Dabei ist zu beachten, daß MS-Netmeeting ab der Version 2.1 eingesetzt wird, da erst ab dieser Version die COM-Schnittstelle verfügbar ist.

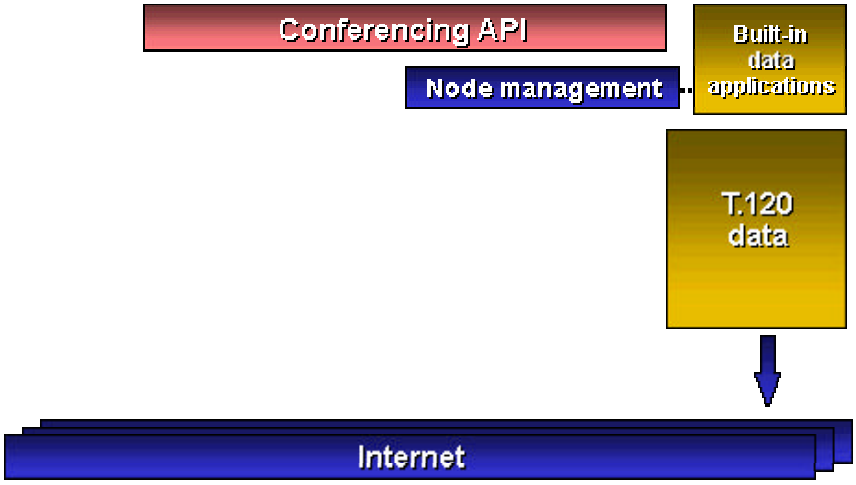



Abbildung 9: Benutzte Netmeeting-Komponenten

**Zugriff auf die Komponenten mittels Component Object Model (COM)**

Der Zugriff auf die Funktionalität und Eigenschaften der Komponenten erfolgt über das *conferencing API*, auf das über das *Component Object Model* (COM) zugegriffen wird. Dieses Modell enthält einen „binären Standard dafür, wie Software-Objekte im Speicher vorliegen, und ein Programmier-Modell, das die Abläufe innerhalb des Objekts versteckt. Das Ganze geschieht sprachenunabhängig, das heißt, daß COM-Objekte mit einer Vielzahl von verschiedenen Programmiersprachen entwickelt werden können." (vgl. ERNST & KOTTLER 1996, S. 304).

Das *Node management* kontrolliert und überwacht den Ablauf der Konferenz bzw. die Teilnehmenden der Konferenz. Über die *Built-in data applications* können Daten zwischen den Teilnehmenden mittels der *T.120 data*-Komponente ausgetauscht werden. Zu diesen Applikationen gehören z. B. Chat, Daten-Übertragung und *whiteboard*. Diese Komponente basiert auf der ITU-T T.120-Empfehlung.

**ITU-T T.120 für die verteilte Applikationsdarstellung**

Zusätzlich ermöglicht die T.120-Komponente auch die verteilte Applikationsdarstellung an alle Teilnehmenden, so daß die geforderte Aufgabenbeobachtung durchgeführt werden kann.

Alle anderen Komponenten konnten aus folgenden Gründen nicht eingesetzt werden:

1. ***Internet Locator Service (ILS)***
   Diese Komponente ist für unser System nicht notwendig, da sich die Konferenzteilnehmenden auf einen gemeinsamen Termin einigen und zu Beginn das System starten. Den Kontakt kann das System dann automatisch direkt herstellen, da die Adressen in einer Datenbank verwaltet werden.[55]

[55] Da wir zu diesem Zeitpunkt davon ausgehen, daß die beteiligten Personen über permanente IP-Adressen verfügen, hat der Einsatz eines ILS keine weiteren Gründe. Erst durch die Berücksichtigung von temporären Adressen wird der Einsatz dieses Service notwendig vgl. Kapitel 7.4.3).

2. ***Call control, Media stream engine (H.263, G.723) , RTP/RTCP (H.225.0)***

Netmeeting keine Verteilung dieser Daten an alle Teilnehmenden be reitstellen konnte. Somit kann auch auf alle weiteren Komponenten verzichtet werden, die sich mit deren Steuerung und Übertragung b - fassen.

Internets (vgl. K 1996). Hiermit definieren sich somit auch die Form des eingesetzten Netzwerks und seine Eigenschaften.

**Netzwerk**

**TCP/IP und Internet**

Mit der Integration von Netmeeting wird das Protokoll des zugrundeliegenden Netzwerks in Form von TCP/IP bestimmt. Dieses Protokoll unterstützt alle Anforderungen, da es durch das Internet weit verbreitet ist und ohne großen Aufwand über diverse Internet Provider für allgemein verfügbar ist. Der Zugang zum Medium Internet kann dabei beliebig erfolgen (z. B. direkter Anschluß, Telefonleitung).

**Multicast-Anbindung**

Die einzige Einschränkung besteht jedoch darin, daß durch den Einsatz der nachfolgenden Audio- und Video-Komponenten ein normaler Internet-Zugang nicht ausreichend ist. Zum Zeitpunkt dieser Arbeit waren keine herkömmlichen Werkzeuge verfügbar, die es erlauben, drei KommunikationspartnerInnen gleichzeitig mit allen anfallenden Audio- und Videodaten zu versorgen. Um dieses zu erreichen, wäre ein spezielles Verteilverfahren der Daten (z. B. Multicast) notwendig, das aber derzeit noch nicht flächendeckend verfügbar ist.[56]

**Audio-Aufzeichnung und -Wiedergabe**

**Soundblaster-kompatible Soundkarte**

Für die Erfassung der Audio-Daten ist es notwendig, daß die eingesetzten Rechner über Soundkarten verfügen, die mit dem Soundblaster-Standard kompatibel sind.

**Zwei Soundkarten für gleichzeitige Übertragung und Aufzeichnung**

Damit gleichzeitig Daten für die Audio-Übertragung und eine Audio-Aufnahme (z. B. Aufzeichnung verbaler Notizen) gespeichert werden können, müssen unter Windows zwei Soundkarten eingesetzt werden. So muß insbesondere der Rechner auf der ExpertInnen-Seite über diese oder eine gleichwertige technische Lösung verfügen, da die ExpertInnen beispielsweise gleichzeitig mit den EntwicklerInnen sprechen (Daten für die Übertragung) und dieses Gespräch für die Dokumentation mitschneiden (Daten für die Dokumentation).

Die externen Geräte zur Audio-Aufnahme (Mikrophon) bzw. -Wiedergabe (Lautsprecher/Kopfhörer) müssen den Anforderungen der gewählten Soundkarten entsprechen.

[56] Wir haben diese Lösung aber trotzdem gewählt, da keine anderen Alternativen einsetzbar waren.

### Audio-Übertragung

**Robust Audio Tool zur Audio-Übertragung**

B. *vat* CuSeeMe, IBM BambaPhone). Eingesetzt haben wir auf Empfehlung des DMN das frei verfügbare Tool (*robust audio tool* *Rat* basiert auf dem Mult - cast-Verfahren und erreicht es dadurch, die anfallenden Audio-Daten an alle *rat* ist es möglich, die EmpfängerInnen der

einen Makel/Whisper-Modus herzustellen. Außerdem ist bei *rat* vorgesehen, eine Steuerung der gesamten Funktionalität von außen zu realisie ren[57] 5.5.4).

**Chipsatz**

Für die Video-Aufzeichnung mit einer externen Videokamera muß in dem eingesetzten Rechner eine Videokarte installiert sein, die aus Leistungsgründen

### Video- und allgemeine Darstellung

**Grafikkarte mit mindestens**

Die darstellende Grafikkarte muß über mindestens 4MB Grafikspeicher ve - fügen, um bei einer Auflösung von 1024 x 768 die geforderte Farbtiefe von

### Video-Übertragung

**Video Conferencing Tool zur Video-Übertragung**

Für die Übertragung der Video-Daten wird das ebenfalls frei verfügbare Tool (*video conferencing tool* *rat* auf dem Multicast-Verfahren *rat* wurde auch für di - ses Tool ein Steuerung von außen vorgesehen.[58]

## 5.2 Konzeptionelle Umsetzung der funktionalen Anforderungen

unserem System zur Unterstützung von Ad-hoc Usability-Tests durch die nachfolgenden Komponenten umgesetzt.

Die Audiokommunikationskomponente stellt einen Audiokanal zur Verfü gung. Insbesondere kann dieser Kanal für die sprachbasierte Verständigung genutzt werden. Die Audiodaten kommen als Stream in das System und kö - nen über eine optionale Audio/Videobearbeitungs-Komponente weiterbea - beitet werden.

Die Videokommunikationskomponente stellt sicher, daß ein direkter visueller Kontakt zwischen den beteiligten KommunikationspartnerInnen hergestellt

In der vorliegenden *rat* angeboten.

[58] *vic* Version, kompiliert für Windows, wird diese Möglichkeit nicht mehr

werden, wie z. B. die Beobachtung der BenutzerIn. Die Videodaten kommen analog zu den Audiodaten als Datenstream und können über die Audio/Videobearbeitung weiterbearbeitet werden.

**Verteilte Applikationsdarstellung**

Mit Hilfe von *shared application* wird ein gemeinsamer, aber sequentieller Zugriff auf eine Anwendung ermöglicht, die auf einem der beteiligten Rechner gestartet worden ist.

**Datentransfer**

Der Datentransfer dient dem Übersenden von Daten jeder Art (z. B. Text, Video, Audio). Für den Datentransfer müssen bestimmte Formate entwickelt werden, um Kombinationen von Daten (z. B. Annotationsdaten) in geeigneter Weise zusammenzustellen.

**E-Mail**

E-Mail ist eine einfache Möglichkeit der asynchronen Kommunikation, die während der eigentlichen Konferenz kaum Anwendung findet. Sie wird hauptsächlich vor oder nach Konferenzen eingesetzt, um weitergehende Informationen zwischen den Teilnehmenden auszutauschen. Diese Funktionalität wird nicht direkt durch das System unterstützt, ist aber im Windows-Betriebssystem standardmäßig verankert (Internet Explorer).

**Datenbasen**

Es werden verschiedene Ausprägungen von Informationssammlungen benötigt. Lokale Datenbanken (z. B. BenutzerIndatenbank für Konferenzkonfigurationen) werden bei allen beteiligten Personen angelegt und von diesen gepflegt.

In zentralen Sammlungen werden alle Daten, die während der Konferenz anfallen, bei der ExpertIn gespeichert. Diese Daten können von allen Teilnehmenden angefordert werden.

**Annotationstool**

Mit diesem Tool ist es möglich, eine oder mehrere multimediale Anmerkungen zu Dokumenten zu erstellen. Diese Annotationen können textueller oder audio-visueller Art sein, wobei die Form des zugrundeliegenden Dokuments beliebig sein kann.

**Konferenzsteuerungskomponente**

Die konkreten Aufgaben dieser Komponente sind sehr umfassend. Alle Aktivitäten, die mit der Steuerung und Kontrolle der eigentlichen Konferenz zusammenhängen, werden mittels dieses Elements be- und verarbeitet. Dazu gehören:

- **Konferenzaufbau**
  Unter Konferenzaufbau, also dem Initialisieren und Aufbau der Konferenz, fassen wir Aktivitäten wie beispielsweise Einladung der Personen, Zustimmung oder Ablehnung einer Einladung zusammen.

- **Konferenzkontrolle**
  Während einer laufenden Konferenz fallen Kontroll- und Steuerung - aufgaben an, wie z.
  Komponente subsumiert werden.
- **Statusanzeigen/Meldungen**
  Über die Kontrolle der Konferenz werden innerhalb dieser Komp - nente alle Meldungen, die mit dem Status der Konferenz zusamme - hängen, verarbeitet und angezeigt.

Über die Einstellungen für Audio und Video hinaus ist es die Aufgabe dieser Komponente festzustellen, von wem die jeweiligen Daten gekommen sind.

Die Daten (z. B. Annotationen), die während einer Konferenz angelegt wu - den, müssen in ihrer zeitlichen Abfolge erkenn- und abrufbar sein. Diese

Zugriff auf die Daten.

**Konferenzmanager**

B.
Namen und Adressen der Beteiligten), werden mittels dieser Komponente

## 5.3

Die Gestaltung der Benutzungsschnittstelle von ErgoNet erfolgt gemäß den Anforderungen, die in Kapitel 4.7 aufgestellt wurden. Die Umsetzung dieser

einem Beispiel erläutern.

- **Etablierung einer hohen Dichte der Funktionalität auf der E pertIn-Seite**
  Die ExpertIn muß während der Testdurchführung die Aktionen der

  Diese Arbeitsaufgaben müssen durch das System bzw. seine Oberflä che optimal unterstützt werden. Im Bereich der Annotation haben wir diesen Aspekt wie folgt umgesetzt:

  für die Annotations-Erstellung

  In unserem System werden die Annotationen komplett mit der Maus
  k-
  e-

  Kommentar-Aufnahme, können durch unterschiedliche Mausaktionen (z.
  nicht von der verteilten Applikation abgelenkt und die Konzentration verbleibt bei der Beobac tung und Analyse der BenutzerIn-Aktionen.

- **Förderung der Selbstbeschreibungsfähigkeit des Systems**
  Die Förderung der Selbstbeschreibungsfähigkeit unseres System haben wir beispielsweise wie folgt verwirklicht:

  ➢ Die Einstellungs-Dialoge sind in verschiedene Bereiche aufgeteilt, die bestimmte Aspekte der Einstellungen des Systems gruppieren. Jeder Bereich ist mit einer Grafik versehen, die mit dem jeweiligen Aspekt assoziiert werden kann. Im folgenden Beispiel verdeutlicht die Grafik, daß sich die nachfolgenden Einstellungen auf Audio-Übertragung innerhalb der Konferenz beziehen.[59]

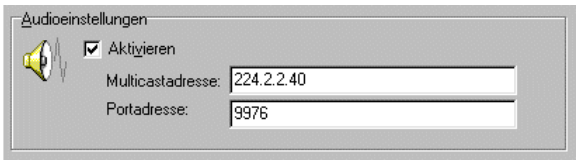



Abbildung 10: Audioeinstellungen

  ➢ Darüber hinaus symbolisieren weitere Grafiken bei Einstellungen, die nur zwischen zwei Werten (z. B. de- und aktiviert) unterscheiden, den aktuell eingestellten Wert. So wechseln die Grafiken zwischen Graustufen- und Farbdarstellung. Zusätzlich stellt ein roter Balken dar, ob die Funktion deaktiviert wurde. Die nachfolgende Grafik verdeutlicht dies, wobei Bild-Aufnahme und Annotation aktiviert und die Audio-Aufnahme deaktiviert ist:


Abbildung 11: Aufnahmeeinstellungen

  ➢ Zusätzlich haben wir den Konferenzaufbau bzw. die Einladung zu einer Konferenz so umgesetzt, daß sie sowohl visuell durch ein Anzeigefenster als auch akustisch durch einen Klingelton angezeigt wird. Den Klingelton haben wir so gewählt, daß er wie ein normales mechanisches Telefonläuten klingt. Dadurch wird die Assoziation mit der Annahme eines Gesprächs ausgelöst, was wiederum die Selbstbeschreibung des Systems unterstützt.

- **Überprüfungen von Eingaben zur Sicherung der Fehlerrobustheit**
  Alle Werte, die durch die Tastatur oder die Maus eingegeben werden, überprüft das System auf Korrektheit. Bei falschen Werten wird auf das korrekte Eingabeformat hingewiesen.

  Beim Eingabeinstrument Maus kann der jeweils zulässige Höchst- bzw. Mindestwert nicht überschritten werden, da die entsprechenden Spin-Buttons deaktiviert werden. Werden die Werte per Tastatur ein-

[59] Lautsprecher mit Frequenzwelle-Icon.

gegeben, so erscheint bei Überschreitung die entsprechende Meldung und der Wert wird auf den maximal zulässigen gesetzt.

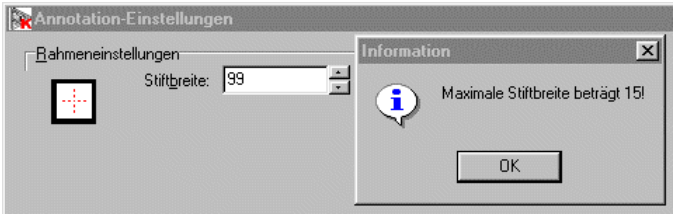



Abbildung 12: Meldung bei fehlerhafter Eingabe

- **Maßnahme zur Umsetzung der Individualisierbarkeit**
  Alle Einstellungen (z. B. Empfang der Annotation, Anzeigedauer der Annotation), die von den jeweiligen BenutzerInnen des Systems eingestellt werden, speichert das System in der Windows-Registry. Beim nächsten Systemstart werden diese Einstellungen wiederverwendet.

## 5.4 Steueraufgaben

**Steueraufgaben sind gering**

Wir haben den Bereich der Steueraufgaben in unserem System auf das Notwendige beschränkt. Auf einem schlanken Basissystem werden mittels schmaler Schnittstellen die einzelnen Module aufgesetzt. Diese Vorgehensweise hat zwei Vorteile:

1. Die einzelnen problemorientierten Module sind auch ohne das Basissystem lauffähig.
2. Es ist leicht möglich, weitere Module in das Basissystem zu integrieren.

**Initialisierungsphase**

Nach dem Start der Applikation findet zunächst die Initialisierungsphase von ErgoNet statt. ErgoNet faßt die Oberflächen der verschiedenen Tools zusammen, um die Anwendung den jeweiligen Benutzenden als ein integriertes Werkzeug anzubieten, unabhängig davon, ob es (im Hintergrund) aus verschiedenen Tools oder Modulen besteht (vgl. Kapitel 3.5.5). In den Anforderungen der Benutzungsoberfläche heißt es (vgl. Kapitel 4.7): „Anwendungen übernehmen immer stärker die Rolle eines Uhrwerks im Hintergrund einer Umgebung, in der sich die Benutzer auf die Manipulation der Daten - im Gegensatz zur Arbeit mit bestimmten Programmen konzentrieren.“. Dieser Gedanke wird in der gewählten Plattform mit dem COM/DCOM-Modell unterstützt.

**COM-Verbindung zu Netmeeting**

Zunächst wird eine COM-Verbindung zu Netmeeting aufgebaut. Netmeeting stellt die grundlegenden Funktionen zum Konferenzaufbau zur Verfügung.


Abbildung 13

In einer weiteren Phase werden allgemeine Konfigurationseinstellungen gela den sowie ein Teil der problemorientierten Module. Die Einstellungen sind hierbei in den jeweiligen Modulen gekapselt.

## Problemorientierte Module

**Umsetzung der Funktionen**

der Anforderungen.

### 5.5.1 Konferenzsteuerung

**Die Konferenzsteuerung**

Wurde die COM-Verbindung zu Netmeeting aufgebaut und alle notwendigen

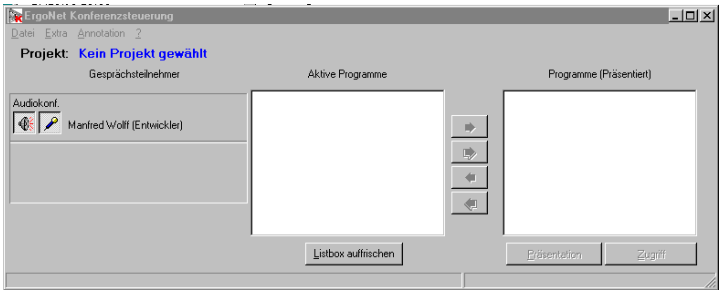



Abbildung 14

Der Dialog „Konferenzsteuerung“ ist in zwei Bereiche aufgeteilt:

1. **Der Bereich der Personenkonferenz**
   Hier finden Statusanzeigen über den Konferenzverlauf statt. Auße - dem können Konferenzkonfigurationen gewählt und die involvierten

2. In diesem Bereich wird angezeigt, welche Programme zum Testen zur Verfügung stehen und welche für andere Personen zum Testen freig - geben werden können (Status ).

Nachdem das System initialisiert wurde, sind zwei weitere Abläufe möglich:

Es wird eine Konferenz initiiert.

Es wird ein Verbindungswunsch angezeigt.

## Konfiguration des Systems

Konfiguration des Systems

Die Konferenzsteuerung wird so konfiguriert, daß eine bestimmte Person, die eine spezifische Rolle beim Usability-Test übernimmt, optimal unterstützt wird. Dadurch kann die Konferenzsteuerung von allen beteiligten Personen gleichermaßen benutzt werden. Der integrative Ansatz von ErgoNet wird an dieser Stelle deutlich. Die Daten (z. B. Multicastadressen der Audio- und Videotools), die normalerweise in unterschiedlicher Form in einzelnen Funktionsmodulen eingegeben werden müssen, werden von ErgoNet in einheitlicher Form erfaßt.

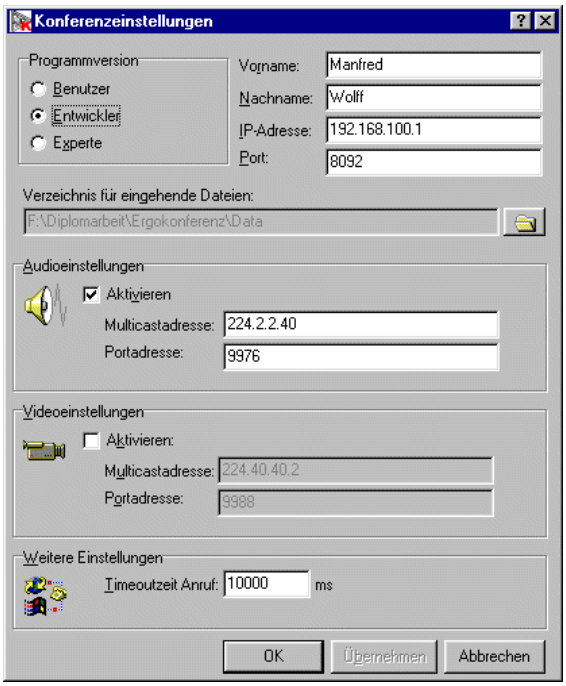
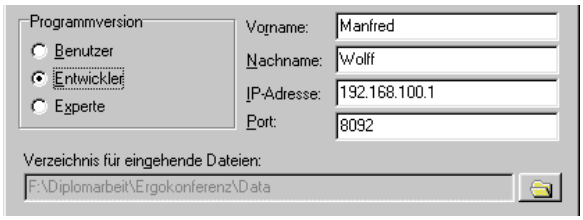



Abbildung 15

Der Konfigurationsdialog ist in drei Ebenen unterteilt:

Konfiguration der Rolle

1. **Eingabe der personenbezogenen Daten mit der jeweiligen Rolle für den Konferenzaufbau mit Netmeeting**




16: Konfiguration der Personeneinstellungen

daß wir Netmeeting als Basissoftware für den Konferenzaufbau ge wählt haben. Notwendige Angaben für den Initialisierungsprozeß mit Netmeeting sind die eigene IP-Adresse und der Vor- bzw. Nachname.

e-

können Voreinstellungen zur Konfiguration des Systems vorgenommen werden.

2. **Konfiguration der Audio- und Videoeinstellungen**

**Audio- und Videoeinstellungen**

Bei unseren Praxistests (vgl. Kapitel 7) haben wir mit Audio- und Videotools im Internet-Multicast Umfeld gearbeitet. Für diese Tools müssen Multicastadressen sowie entsprechende Portadressen konfiguriert werden. Die Audio- und Videounterstützung kann individuell ein- bzw. ausgeschaltet werden. Wenn wie in diesem Beispiel die Audioeinstellung aktiviert wird, werden im Bereich der Konferenzsteuerung spezifische Einstellungen für Mikrophon und Lautsprecher sichtbar.

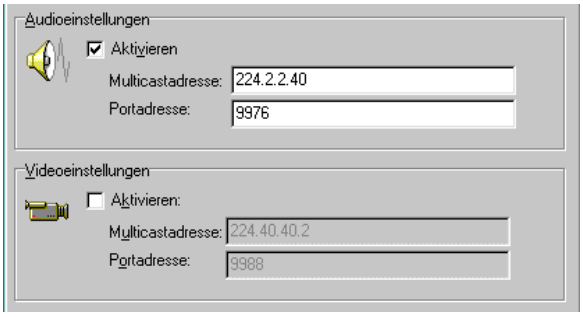



Abbildung 17: Konfiguration der Audio- und Videoeinstellungen

3. **Weitere Einstellungen**

**Weitere Einstellungen**

Im dritten Bereich werden weitere Einstellungen konfiguriert. Das Feld „Timeoutzeit Anruf“ gibt an, wie lange ein Verbindungsaufbau dauern darf, d. h. nach wieviel Millisekunden die angerufene Person reagiert haben muß. Wenn nach Ablauf dieser Zeit ein Anruf weder angenommen noch abgelehnt wurde, wird eine Fehlermeldung ausgegeben.

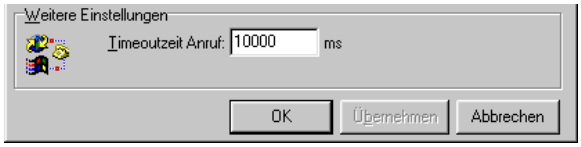



Abbildung 18: Weitere Einstellungen

**Einstellung für den Empfang der entfernten Kontextmarkierungen**

Zusätzlich kann eingestellt werden, ob die Anzeige der Kontextmarkierungen, die von der ExpertIn erstellt und an alle Teilnehmenden versendet wird, angezeigt werden soll. Da dies keine grundsätzliche Einstellung ist, sondern im Verlauf der Konferenz evtl. wiederholt verändert wird, ist dieser Punkt in das Menü „Annotationen“ unter dem Eintrag „Empfangen“ integriert worden. Durch die Auswahl dieses Menüpunkts wird der Empfang abwechselnd de- bzw. aktiviert.

## Konferenzaufbau

**Aktiver Konferenzaufbau**

Wie oben beschrieben, ist ein aktiver und ein passiver Konferenzaufbau möglich. Der aktive Konferenzaufbau hat folgende Phasen:

1. **Auswahl der Konferenzkonfiguration**

**Auswahl der Konferenzkonfiguration**

Aus den vorgegebenen Konferenzkonfigurationen wird eine ausgewählt. Die Konferenzkonfiguration wird zu Beginn eines Projekts erstellt und an alle Teilnehmenden verschickt (vgl. Kapitel 4.2).


Abbildung 19: Aktiver Konferenzaufbau

**Status der beteiligten Personen**

Anhand der Konferenzkonfigurationsdaten wird ermittelt, welche Personen an der Konferenz teilnehmen diese werden angezeigt.


Abbildung 20: Konferenzetablierung

An den Statusanzeigen, die den Namen zugeordnet sind, ist erkennbar, zu welchen Personen ein Konferenzaufbau möglich ist. Die Benutzerin hat im obigen Beispiel bereits ihre Konferenzsteuerung aktiviert (Anzeige des Namens ist schwarz), während der Experte noch offline ist (Anzeige des Namens ist grau). Zur Benutzerin kann aus diesem Grund eine Verbindung aufgebaut werden.

### 2. Aufbau der Verbindung

**Aufbau der Verbindung**

Mit Hilfe des Menüs oder der Schnellschaltfläche (*speedbutton)*  kann eine Person zu der Konferenz eingeladen werden. Wird die Einladung akzeptiert, dann wechselt der Status der Schaltfläche vom Telefon in einen grünen, eingerasteten Schalter.


Abbildung 21: Konferenzstatus nach Konferenzaufbau

**Kontextmenü Eigenschaften als Verweis auf die BenutzerIndatenbank**

Durch einen Klick mit der rechten Maustaste auf den Schalter, der der jeweiligen Person zugeordnet ist, kann über das Kontextmenü „Eigenschaften“ zur BenutzerIndatenbank (vgl. Kapitel 5.5.3) verzweigt werden. Der entsprechende Eintrag in der Datenbank wird angezeigt, wobei aber keine Veränderungen an diesem Eintrag bzw. den anderen Einträgen vorgenommen werden können.

**Passiver Konferenzaufbau**

Beim passiven Konferenzaufbau wird die jeweilige Person über einen Aufbauwunsch informiert.

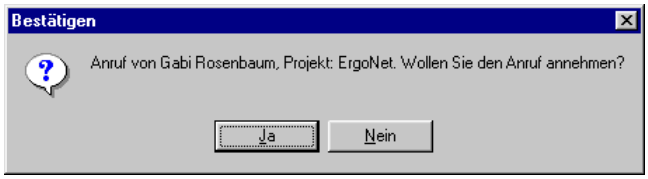



Abbildung 22: Passiver Konferenzaufbau

Wird die Verbindung akzeptiert, präsentiert sich das gleiche Bild wie Abbildung 21. Der Online/Offline-Status der Beteiligten wird alle 30 Sekunden aufgefrischt. Sobald die ExpertIn die Konferenzsteuerung gestartet hat, wird der Name ebenfalls schwarz dargestellt und kann nun zur Konferenz dazustoßen.

**Starten von Audio und Video**

Mit der Etablierung der Konferenz via Netmeeting können die Audio- und Videotools gestartet werden, um eine visuelle bzw. Sprachverbindung zu den Teilnehmenden aufzubauen.

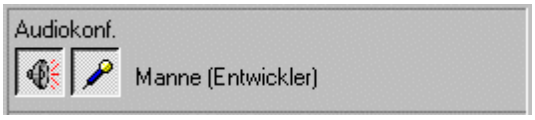



Abbildung 23: Audiostatus

**Mixerkomponente**

Die in der Anforderungsermittlung angesprochene Mixerkomponente wurde in diesem Prototyp nur sehr rudimentär implementiert. Wurde wie in diesem Beispiel die Audiokomponente aktiviert, werden die Kontrollen für die eigene Audioaufnahme und -wiedergabe dargestellt. Über diese Schnellschaltflächen ist es problemlos möglich, den eigenen Ton wegzuschalten bzw. den einkommenden Audiostream auszuschalten.

**Verbindung abbauen**

Einzelne Personen können zu der Konferenz eingeladen werden. Eine Ausladung ist nicht vorgesehen. Personen können über das Menü aus der Konferenz austreten und später wieder dazustoßen. So können z. B. EntwicklerInnen, die nicht immer ein Interesse an der Verfolgung des Gesamttests haben, die Konferenz verlassen und eventuell zu einem späteren Zeitpunkt wieder der Konferenz beitreten.


Abbildung 24: Konferenz beenden

### 5.5.2 Shared Application

**Verteile Applikations-darstellung**

Die verteilte Applikationsdarstellung mittels s*hared application* ist ein Kernbestandteil des Gesamtsystems. Diese Funktionalität wird für mehrere Bereiche benötigt:

- Verfolgung der Durchführung der Arbeitsaufgabe durch die EntwicklerIn und die ExpertIn.
- Möglichkeit, während des Tests steuernd und kontrollierend eingreifen zu können.
- Durch das Sharen von Applikationen ist eine Zusammenarbeit über das Netz möglich.

**Fast jede Anwendung kann geshared werden**

Bis auf einige Ausnahmen kann jede Applikation durch *application sharing* anderen Personen zugänglich gemacht werden. Die Ausnahmen sind vor allem Programme, die eine Beziehung zu Netmeeting haben, wie z. B. *whiteboard*[60]. Auch die Konferenzsteuerung ErgoNet ist aus diesem Mechanismus ausgeschlossen.


Abbildung 25: Aktive Programme anzeigen

**Aktive Programme**

In der Listbox „Aktive Programme“ sind die Programme aufgelistet, die anderen Personen während der Konferenz zugänglich gemacht werden können. Wird der Schalter „Listbox auffrischen“ betätigt, wird die Liste der aktiven Programme erneuert.

#### Programme präsentieren

**Präsentierte Programme**

Soll ein Programm anderen Personen visuell zugänglich gemacht werden, so muß dieses Programm präsentiert werden.

[60] Die Stärke derartiger Anwendungen liegt darin, daß sie nicht geshared werden, sondern getrennt an allen Enden der Konferenz gleichartig vorhanden sind. Somit ist nur ein Nutzdatenaustausch erforderlich.

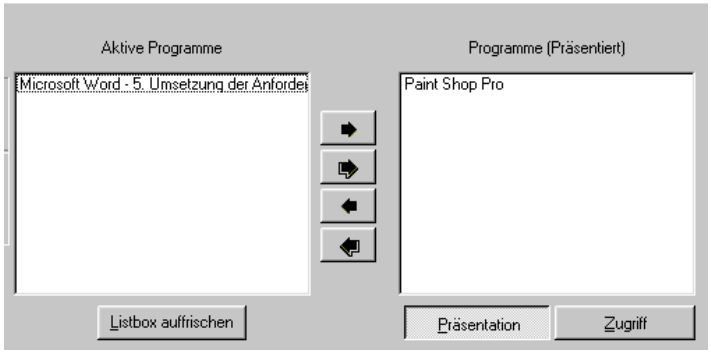



Abbildung 26: Programme präsentieren

Dazu wird es mit Hilfe der Pfeiltasten aus der Liste der „Aktiven Programme" in die Liste der präsentierten Programme verschoben. Alle Applikationen, die in der Listbox der präsentierten Programme auftauchen, werden von allen Teilnehmenden gesehen.

### Zugriff ermöglichen

**Zugriff geben und nehmen**

Damit andere ebenfalls mit dem Programm arbeiten können, muß sowohl Zugriff gewährt werden (von der Personen, die das Programm präsentiert) als auch Zugriff genommen werden (von der Person, die auch mit dem Programm arbeiten will). Diese doppelte Semantik des Schalters Zugriff haben wir von Netmeeting übernommen.[61]

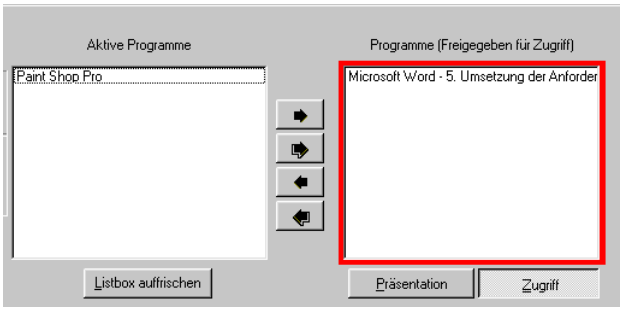



Abbildung 27: Auf Programme zugreifen

**Einschränkung des application sharing**

Im obigen Beispiel wird das Programm Microsoft Word für alle zum Zugriff freigegeben. Andere Personen sehen das Programm auf ihrem Bildschirm und können mit dem Programm so arbeiten, als wenn es auf ihrem Rechner gestartet worden wäre. Die einzige Einschränkung des *application sharing* ist, daß immer nur eine Person zur gleichen Zeit mit dem Programm arbeiten kann. Ein freigegebenes Programm hat über der Fensterzeile eine Statusinformation. Aus dieser Information kann ersehen werden, wer das Programm freigegeben hat. Außerdem sind die verschiedenen Cursors der anderen Personen zu sehen.

[61] Nach unserer Auffassung würde es NutzerInnen, die gelegentlich Netmeeting „pur" benutzen, mehr verwirren, würde hier eine „sprechendere" Bezeichnung eingeführt. Wir bewerten hier Konsistenz höher als verbesserte Selbsterklärungsfähigkeit.

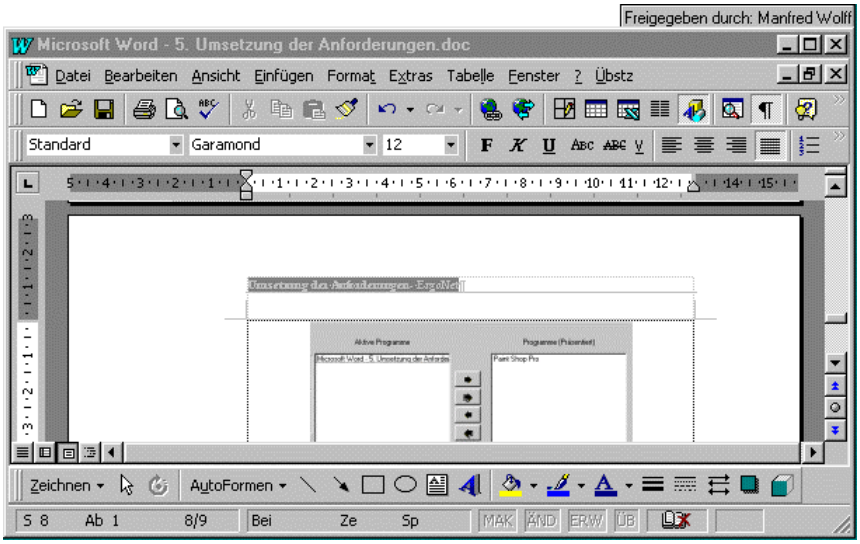



Abbildung 28: Freigegebene Applikation

**Systeminterne Behandlung**

Unter Windows ist es möglich, gesharte Fenster zu identifizieren und Aufzeichnungen über die verschiedenen Bewegungen der entfernten BenutzerIn zu machen.

### 5.5.3 BenutzerIndatenbank

**BenutzerInkonfiguration**

ErgoNet kann verschiedene BenutzerInkonfigurationen erstellen und pflegen. Eine Konfiguration besteht aus:

- Einem Konferenzgegenstand, meist ein Softwareprojekt.
- Drei Personen: EntwicklerIn, ExpertIn und BenutzerIn.
- Einem Prefix für die Annotation (vgl. Kapitel 5.5.6).
- Optionale Bemerkungen.

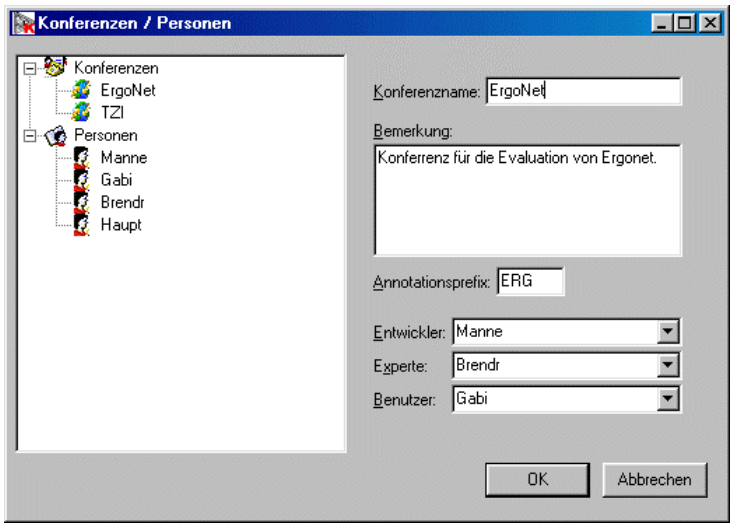



Abbildung 29: Konferenzkonfigurationen erstellen und editieren

**Dialog wechselt im Kontext**

Konferenzkonfigurationen und Personen werden mit Hilfe des gleichen Dialogs eingerichtet. Je nach Kontext (Konferenz oder Person) wechseln die Eingabemöglichkeiten auf der rechten Dialogseite.

**Objektorientierte Baumdarstellung**

Die objektorientierte Baumdarstellung ermöglicht eine schnelle Übersicht über Konferenzen und Personen. Denkbar ist, daß im weiteren Ausbau der Software weitere Objekte angezeigt und editiert werden können.

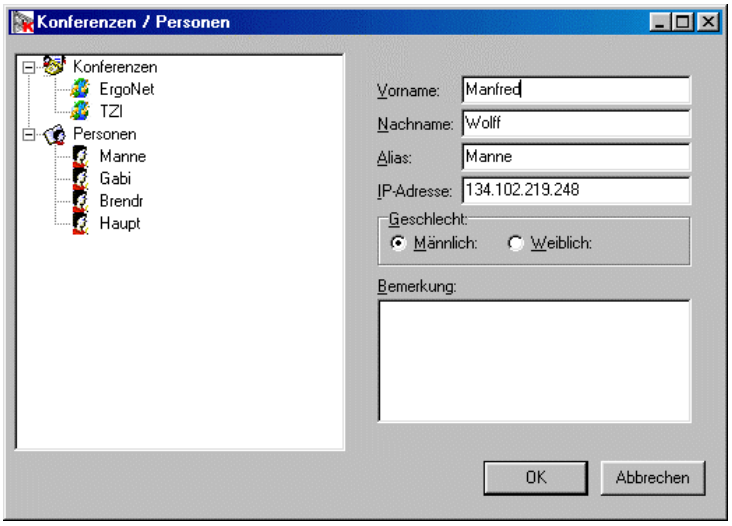



Abbildung 30:KonferenzteilnehmerInnen erstellen und editieren

**Personeneinstellungen**

Neben den Einstellungen, die von Netmeeting gefordert werden (Vor-, Nachname und IP-Adresse), gibt es bei der Eingabe der Personen weitere Einstellmöglichkeiten:

- Über den Alias können eindeutige Namen in Form von Spitznamen eingegeben werden. Diese Namen erscheinen statt des Vor- und Nachnamens in der Konferenzsteuerung.
- Die Eingabe des Geschlechts ermöglicht die geschlechtsspezifische Anrede während der Konferenzsteuerung (Benutzerin vs. Benutzer).
- Das Bemerkungsfeld ist wie bei der Konferenzkonfiguration optional und kann frei benutzt werden.

### 5.5.4 Audio- und Videoübertragung

**MBone-Tools**

Die Werkzeuge zur Audio- und Videoübertragung *rat* und *vic* kommen aus dem Internet-Multicast-Umfeld.

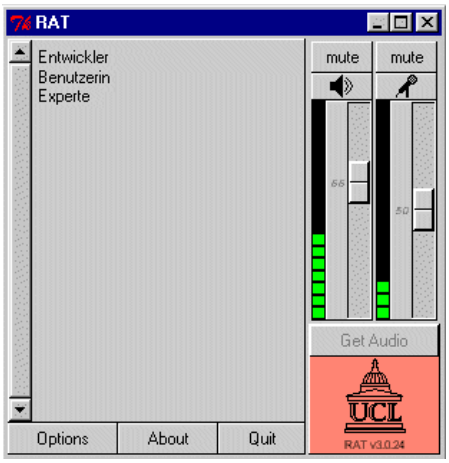



Abbildung 31: Audiotool (*rat*)

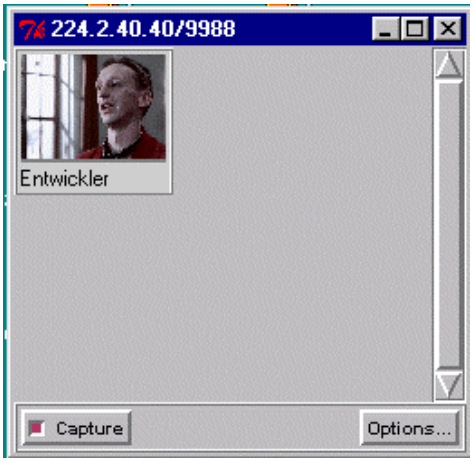



Abbildung 32: Videotool (*vic*)

Wir entschieden uns für diese Werkzeuge, da zum Zeitpunkt der Umsetzung keine anderen die in Kapitel 4 aufgestellten Anforderungen an die Audio- und Videoübertragung erfüllten. Insbesondere waren andere Tools nicht in der Lage, verschiedene Datenströme gleichzeitig zu verschiedenen Adressaten zu senden, um eine synchrone Dreipunktkommunikation zu realisieren.

**Keine detaillierte Beschreibung der Tools**

Wir verzichten auf eine detaillierte Beschreibung der eingesetzten Tools, weil wir davon ausgehen, daß sie nur temporär eingesetzt werden können. Wir begründen dieses wie folgt:

1. Die Werkzeuge sind nicht nach dem Windows-Styleguide konzipiert. Insbesondere auf der Seite der BenutzerIn erfordert der Einsatz dieser Tools ein Umdenken, weil sowohl Erscheinungsbild als auch Benutzung nicht gängigen Abläufen entsprechen. Anhand der nachfolgenden Abbildung verdeutlichen wir dies an einigen Beispielen.

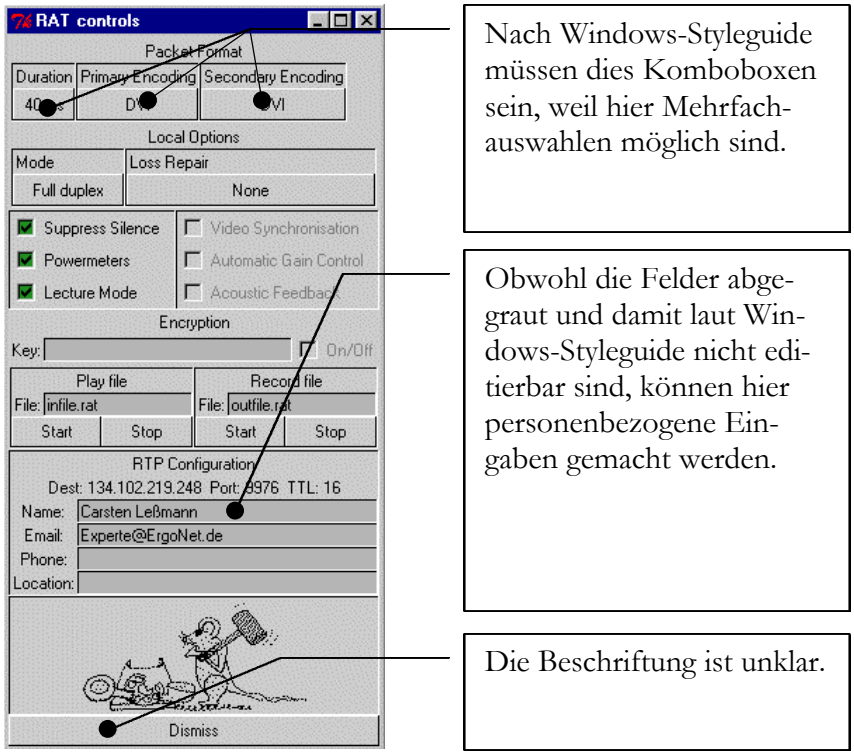



Abbildung 33: Konfiguration *rat*

2. Die Tools bieten keine Möglichkeit zur Steuerung, so daß die Funktionalität nicht in die Benutzungsoberfläche integriert werden kann (vgl. Kapitel 4.7.2).
3. Die notwendige Technik (Multicast) ist nicht allgemein verfügbar.

### 5.5.5 Datenübertragung

**Verschiedene Möglichkeiten der Datenübertragung**

Für die Übertragung von Daten kommen durch die gewählte Netztopologie mehrere Medien in Betracht:

- Übertragung durch einen eigenen TCP-Kanal.
- Datenübertragung durch Internetdienste, z. B. FTP.
- Datenübertragung durch den T.120-Kanal von Netmeeting.

**Entscheidung zugunsten T120**

Wir haben uns für die letzte Möglichkeit entschieden, weil die Konferenzsteuerung sehr stark mit Netmeeting verwoben ist. So brauchten wir nicht noch einen weiteren Dienst, wie z. B. FTP, zu implementieren und konnten vorhandene Schnittstellen benutzen.

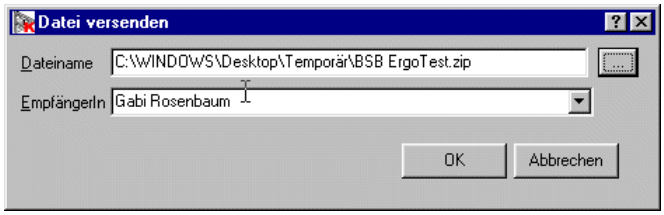



Abbildung 34: Dateiname und EmpfängerIn eingeben

Die Netmeeting-Datenübertragungs-Schnittstelle bietet an, die Datei entweder an eine oder an alle Personen zu übertragen. Außerdem wird der Dateiname der zu übertragenen Datei benötigt und über einen Dialog eingestellt.

Mit Hilfe der Windows-Animationen wird der Fortschritt des Dateitransfers angezeigt.

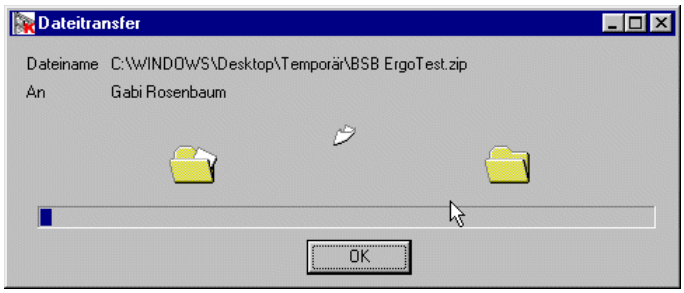



Abbildung 35: Fortschritt Dateiübertragung

### 5.5.6 Annotation

**Parallele Dokumentation durch Erstellung von Annotationen**

Die Dokumentation muß parallel zur Durchführung des Tests angefertigt werden. Aus diesem Grund setzen wir das Prinzip der Annotation ein. Während des Tests werden Screenshots des Testgegenstands angefertigt, die mit verbalen Kommentaren der ExpertIn kombiniert werden.

Hiermit können parallel zum eigentlichen Test Dokumente hergestellt und von den EntwicklerInnen anschließend benutzt werden, um beispielsweise den Testgegenstand zu modifizieren und dadurch zu verbessern.

**Kontextmarkierungen**

Die ExpertIn kann über das eigentliche Bild des Screenshots hinaus Bereiche auf diesem Bild markieren, um das Augenmerk auf bestimmte Kontexte des Testgegenstands zu lenken, die wir im weiteren Kontextbereiche nennen. Hierbei stehen die folgenden Techniken zur Verfügung:

**Fixierter Rahmen**

1. Ein rechteckiger Rahmen, der in seinen Maßen fixiert ist, d. h., die Größe wird vor Beginn der Annotation festgelegt. So ist es beispielsweise möglich, wiederkehrende geschlossene Bereiche der Oberfläche (z. B. Buttons) schnell und einfach zu markieren.

**Flexibler Rahmen**

2. Ein rechteckiger Rahmen, der während der Annotationserstellung in seinen Ausmaßen verändert werden kann. Mittels dieses flexiblen Rahmens können unterschiedlich große geschlossene Bereiche schnell und einfach hervorgehoben werden.

**Pfeile als Verweise**

3. Ein Pfeil, der während der Annotationserstellung in seiner Zeigerichtung verändert werden kann. Hierbei wird kein geschlossener Bereich markiert, sondern auf bestimmte Merkmale der Oberfläche (z. B. ein Icon) verwiesen.

**Lokale und entfernte Anzeige der Kontextmarkierungen**

Die Kontextmarkierungen werden bei allen TeilnehmerInnen an der entsprechenden Position, an der die ExpertIn sie erstellte, angezeigt. Die Anzeige der Markierung und Dauer der Anzeige kann individuell eingestellt werden.

**Beibehalten des Kontextes während der Anzeige der Kontextmarkierung**

Während der Anzeige der Kontextmarkierung wird der Neuaufbau des jeweiligen Fensters verhindert. Dies bedeutet konkret, daß das zum Zeitpunkt der

Kontextmarkierung dargestellte Bild für die gesamte Darstellungsdauer der Markierung sichtbar bleibt.

**Definition von Annotation**

Diese Kombination aus Screenshot, Kontextmarkierung und dem verbalen Kommentar nennen wir im weiteren Annotation, die eine einfache und parallele Erstellung einer Dokumentation durch die ExpertIn ermöglicht.

**Aktivierung der Annotations-Erstellung**

Die Erstellung der Annotationen, bei dem das gesamte System in den Annotationsmodus wechselt, wird durch Auswahl des Menüpunktes „Annotation/Starten...“ in der Konferenzsteuerung aktiviert. Vor dem tatsächlichen Beginn wird ein Einstellungs-Dialog angezeigt. Die Aktivierung dieses Menüpunkts ist an keine Voraussetzungen gebunden[62], d. h., es ist prinzipiell möglich, zu jedem Zeitpunkt, also auch ohne eine konkrete Konferenzsituation, Annotationen zu erstellen. Die ExpertIn wäre dann beispielsweise in der Lage, im Rahmen möglicher Nacharbeiten weitere Annotationen zu bereits vorhandenen Annotationen oder anderen Dokumenten zu erstellen.

### Konfiguration der Annotation

**Speicherung der Einstellung in der Windows-Registry**

Bevor mit der Annotationserstellung begonnen werden kann, müssen bestimmte Einstellungen für das Erscheinungsbild der Annotationen durchgeführt werden. Die Werte dieser Einstellungen werden in der Windows-Registry (vgl. Kapitel 5.6) gespeichert. Wir unterscheiden zwischen zwei verschiedenen Arten der Einstellungen, die die Erstellung der Dokumentation effizienter gestalten.

**Unterscheidung der Einstellungen**

1. **Grundsätzliche Einstellungen**
   Diese Einstellungen sollten vor dem eigentlichen Beginn der Annotationserstellung gemacht werden. In dieser Kategorie haben wir die Einstellungen zusammengefaßt, die sich mit dem Erscheinungsbild der Rahmen (Rahmeneinstellungen), den notwendigen Zeitmaßen (Anzeigedauer der Kontextmarkierung und maximale Aufnahmedauer des Kommentars) und dem Speichern der Annotationen (Pfad) befassen.

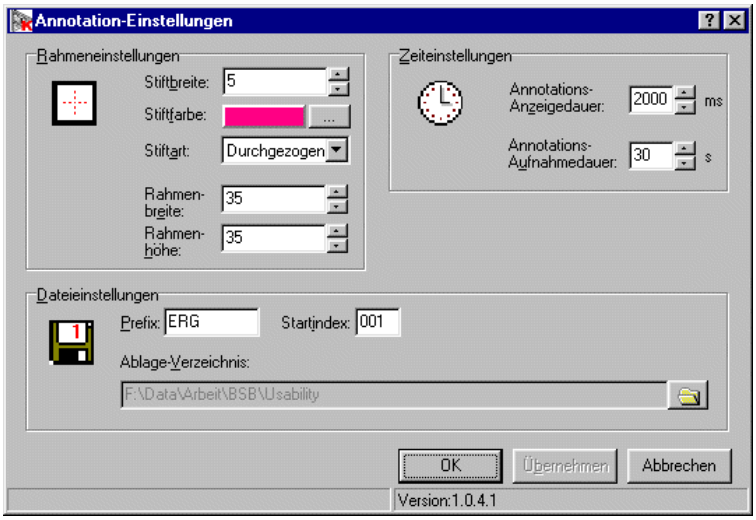



Abbildung 36: Grundsätzliche Annotations-Einstellungen

[62] Eine mögliche Voraussetzung wäre beispielsweise, daß eine verteilte Applikation sichtbar sein muß.

**Aufteilung in drei Bereiche**

Der Dialog der grundsätzlichen Annotationseinstellungen unterscheidet die folgenden drei Bereiche:

**Einstellungen für den Rahmen als Kontextmarkierung**

a) Rahmeneinstellungen
Dieser Teil des Dialogs ermöglicht es, die Stiftbreite des Rahmens, der in der Annotation als Kontextmarkierung dargestellt wird, einzustellen.

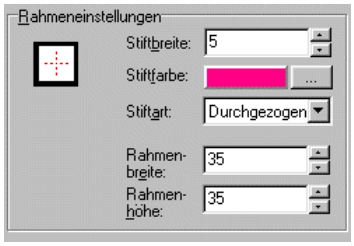



Abbildung 37: Rahmeneinstellungen

**Linienstärke, Linienfarbe und Linienart**

Die Werte[63] dürfen zwischen eins (Haarlinie) und fünfzehn (dicke Linie) liegen, wobei die Linienfarbe in einem Standard-Farbdialog[64] eingestellt werden kann. Die Farbe wird bei der Anzeige der Kontextmarkierung mit der aktuellen Hintergrundfarbe kombiniert, um keine Informationen durch die Markierung selber zu verdecken. Zusätzlich kann entschieden werden, welcher Art die Linie sein soll. Die Drop-Down-Box bietet folgende Möglichkeiten:

- Durchgezogen
  Die Linien des Rahmens werden komplett durchgezogen gezeichnet, wobei alle Linienstärken zulässig sind. Sollte die Einstellmöglichkeit der Linienstärke vorher deaktiviert sein, so wird sie durch die Auswahl dieser Linienart wieder aktiviert.

- Gestrichelt, Gepunktet
  Bei der Auswahl dieser Linienarten wird die Einstellmöglichkeit der Linienstärke deaktiviert und die Stärke auf eins fixiert.[65]

[63] Alle Angaben von Größen erfolgen in Pixeln.

[64] Mit Standard-Farbdialog ist in diesem Zusammenhang der Dialog gemeint, den das Windows-System allgemein bereitstellt.

[65] Dies begründet sich mit der Einschränkung des Windows-API auf diesen Wert (vgl. WIN32API 1996).

**Ausmaße des fixierten Rahmens**

Die bisherigen Einstellungen betreffen sowohl den fixierten Rahmen als auch den flexiblen. Die Einstellung der Größe des Rahmens in Höhe und Breite betrifft nur den fixierten Rahmen. Die maximalen Werte richten sich nach den Werten der aktuellen Auflösung, die nicht überschritten werden können. Bei falscher Rahmenbreite wird folgende Meldung abgesetzt:


Abbildung 38: Meldung bei falscher Rahmenbreite

**Einstellungen für die Anzeigedauer der Kontextmarkierung und Dauer der Aufnahme der Kommentar-Datei**

b) Zeiteinstellungen
Mittels dieses Teils des Einstellungsdialogs werden die Werte für die Anzeigedauer der Kontextmarkierung (in Millisekunden) und die maximale Aufnahmedauer des Kommentars (in Sekunden) bestimmt.

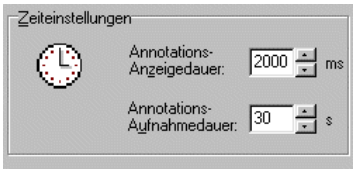



Abbildung 39: Zeiteinstellungen

**Maximale Werte**

Die Anzeigedauer der Annotation bezieht sich auch auf die Dauer der Anzeige der empfangenen Annotationen der ExpertIn. Die maximalen Werte liegen bei 5000 ms für die Anzeigedauer und 120 s für die Aufnahmedauer. Analog zu den Rahmeneinstellungen wird eine angepaßte Meldung bei Überschreitung dieser Werte angezeigt.

**Einstellungen für die Speicherung der Annotationen**

c) Dateieinstellungen
In diesem Teil des Dialogs werden die Einstellungen vorgenommen, die sich mit der Persistenz und Ordnung der erstellten Annotation beschäftigen.

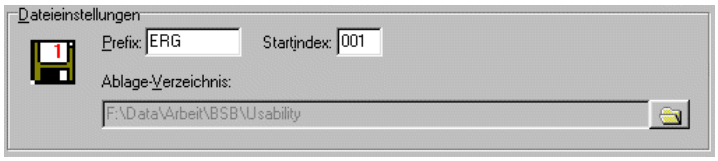



Abbildung 40: Dateieinstellungen

**Prefix als Strukturierungselement**

Mit dem Prefix besteht die Möglichkeit, Annotationen bestimmten Konferenzen (Prefix als Zuordnung zur Konferenz) bzw. bestimmten Teilen der Arbeitsaufgabe einer Konferenz (Prefix als Zuordnung zur Arbeitsaufgabe) zuzuordnen. Die maximale Länge des Prefix ist auf drei alphanumerische[66] Zeichen begrenzt und ist Teil des Dateinamens.

**Kombination von Prefix und Index zu eindeutigen Dateinamen**

Der Startindex, der in Kombination mit dem Prefix verwendet wird, besteht aus drei numerischen Zeichen. Dieser Index wird mit dem angegeben Startwert bei jeder neu erstellten Annotation inkrementiert und zusammen mit dem Prefix als Dateiname der aktuellen Annotation eingesetzt.

**Pfadangabe für den Speicherort**

Die Annotationen werden in dem angegebenen Verzeichnis gespeichert, das mit einem Standard-Dateidialog gewählt und gegebenenfalls neu angelegt werden kann.

**Einstellungen für individuelle Annotationen**

2. **Situationsbedingte Einstellungen**

Während bzw. vor der Annotationserstellung können diese Einstellungen verändert werden, um auf verschiedene Anforderungen des Testgegenstands zu reagieren. Hierzu gehören Einstellungen, die die Ausprägung der Annotation (Screenshot, Kontextmarkierung, Kommentar-Aufnahmen), die Art der Verweismarkierung (Pfeilrichtung und Größe) und die Position des Statusfensters der Annotation betreffen.

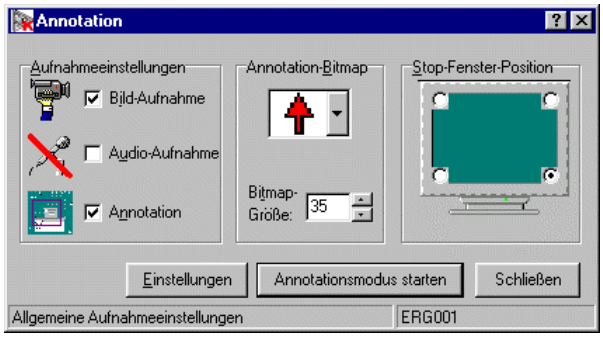



Abbildung 41: Weitere Annotations-Einstellungen

**Aufteilung in drei Bereiche**

Dieser Dialog besteht aus den folgenden drei Bereichen:

**Einstellungen der Inhalte einer Annotation**

a) Aufnahmeeinstellungen

Die Veränderung dieser Einstellungen bestimmt die inhaltliche Zusammensetzung einer Annotation, wobei die jeweilige Einstellung durch das zugeordnete Icon visuell unterstützt wird. Bei Aktivierung wird ein farbiges Icon angezeigt, daß mit der Art der Einstellung assoziiert ist. Die Deaktivierung verursacht die Anzeige des gleichen Icons in graustufiger Darstellung mit einem roten Balken.

**Screenshot-Erstellung**

Die Bildaufnahme kann deaktiviert bzw. aktiviert werden. Bei einer deaktivierten Aufnahme wird die Kontextmarkierung trotzdem angezeigt und versendet.

[66] In diesem Zusammenhang alle alphanumerischen Zeichen, die das Windows-System als gültige Zeichen für einen Dateinamen vorsieht (vgl. WIN32API 1996).

Der Erstellung einer Kommentar-Datei kann mittels der Audio-Aufnahme-Einstellung erfolgen bzw. unterbunden werden.

**Aufnahme einer Kommentar-Datei**

Die Anzeige und Versendung einer Kontextmarkierung wird durch die Veränderung der Annotations-Einstellung bestimmt. Da diese Kontextmarkierung grundlegender Bestandteil einer Annotation ist, bedeutet die Deaktivierung dieser Einstellung praktisch nur die Erstellung eines Screenshots ohne Kontextmarkierungen.

**Erstellung der Kontextmarkierung**

b) Annotation-Bitmap
Über den Einsatz von Rahmen als Kontextmarkierung hinaus besteht die Möglichkeit, Pfeile als Verweise auf bestimmte Bereiche der Oberfläche des Testgegenstands einzusetzen. In diesem Teil kann ein Standard-Pfeil ausgewählt werden, der während der Annotations-Erstellung bei der Nutzung der rechten Maustaste eingesetzt wird. Darüber hinaus kann die Größe des Pfeils eingestellt und verändert werden. Die maximale Größe des Pfeils ist auf 300 Pixel beschränkt, da ansonsten die anderen Bereiche komplett überdeckt würden.

**Bitmap als Verweis auf einen Bereich**

c) Stop-Fenster-Position
Während der Annotations-Erstellung zeigt ein kleines, dauernd sichtbares Statusfenster den aktuellen Zustand im Annotationsmodus an. Mittels dieser Einstellung kann bestimmt werden, an welcher Bildschirmposition dieses Fenster während der Annotations-Erstellung angezeigt werden soll. Die Auswahl ist auf die jeweiligen vier Ecken des Bildschirms beschränkt. So kann während der Erstellung der Annotationen verhindert werden, daß dieses Statusfenster andere Informationen auf dem Bildschirm, wie z. B. die verteilte Applikation, überdeckt.

**Position des Statusfensters**

Da dieser Dialog vor der Erstellung der Annotation angezeigt wird, kann von hieraus direkt zu den grundsätzlichen Einstellungen gewechselt werden, die während des Tests ohne großen Aufwand verändert werden können.

**Einstellungen für alle und individuelle Annotationen**

Wenn alle Einstellungen vorgenommen worden sind, kann mit dem Beginn der Erstellung der Annotationen begonnen werden.

### Beginn der Annotation

Bei der Erstellung der Annotation wird zwischen dem Normal- und Annotationsmodus unterschieden. Der Normalmodus bezieht sich auf den Zustand und das allgemeine Systemverhalten von Windows (z. B. Mausfunktionalität, Aktivierungsverhalten von Objekten), wie es bei einer Standard-Nutzung vorgesehen ist. Für die Erstellung der Annotationen ist es notwendig, in den sogenannten Annotationsmodus umzuschalten.

**Normal- und Annotationsmodus**

Diese Umschaltung greift umfassend in die Arbeits- und Funktionsweise des Windows-Systems ein. Alle Mausaktionen werden während des Annotationsmodus abgefangen, an ErgoNet umgeleitet und hier weiterverarbeitet. Dies führt dazu, daß eine herkömmliche Bedienung von Windows im Annotationsmodus nicht möglich ist.

**Eingriff in das Windows-System**

**Vorgeschalteter Einstellungsdialog**

Der vorgeschaltete Einstellungsdialog wird durch die Aktivierung des Annotationsmodus geschlossen, und das Statusfenster der Annotation erscheint an der gewünschten Position.

Der Zustand im Annotationsmodus wird mittels des Statusfensters angezeigt. Dieser Fenster ist in folgende Bereiche unterteilt:

**Titelzeile mit Statustext**

1. **Titelzeile**
   Der Text der Titelzeile beschreibt den Status, in dem sich das Windows- und ErgoNet-System im Rahmen des Annotationsmodus befindet.

**Grafik als Statusanzeige**

2. **Statusgrafik**
   Die Statusanzeige der Titelzeile wird durch eine Grafik im Statusfenster dadurch visuell unterstützt, daß sie mit der jeweiligen Aktion assoziiert werden kann.


Abbildung 42: Statusfenster im Annotationsmodus

**Button, um den Annotationsmodus zu verlassen**

Der STOP-Button beendet den Annotationsmodus und stellt die herkömmliche Funktionsweise des Windows-Systems wieder her. Der Einstellungsdialog wird erneut angezeigt.

### Verschiedene Status während der Erstellung der Annotation

**Statusunterscheidung während der Annotations-Erstellung**

Die Erstellung einer kompletten Annotation verläuft in drei aufeinanderfolgenden Schritten, die durch das Statusfenster angezeigt werden. Zusätzlich werden die unterschiedlichen Stadien der Annotationserstellung durch angepaßte Maus-Cursor-Formen visualisiert.

Im folgenden werden die Schritte einzeln beschrieben. Die jeweiligen Formen des Maus-Cursors und das Aussehen des Statusfensters wird abschließend in einer Tabelle dargestellt.

**Bereitschafts-Status**

1. **Bereit**
   In diesem Zustand wartet das Ergo-Net-System im Annotationsmodus auf den Beginn der Erstellung einer neuen Annotation. Durch einen Mausklick wird ein Screenshot des Fensters erstellt, in dem sich der Maus-Cursor während des Klicks befindet. Abhängig von der gewählten Maustaste wird die Art der Kontextmarkierung bestimmt.

   In Abhängigkeit der gewählten Markierungsart geht das ErgoNet-System in den nächsten Zustand über. In diesem Zustand wird die Kontextmarkierung an der Position erstellt bzw. begonnen, an der der Mausklick stattfand.

**Erstellung eines flexiblen Rahmens**

2. **Flexibler Rahmen**
   Der Beginn der Erstellung eines flexiblen Rahmens wird durch den Einsatz der linken Maustaste gestartet. Anhand der Maus-Cursor-Bewegungen wird ein Rahmen mit gepunkteter Linie in der eingestellten Farbe vorgezeichnet, damit dieser Vorgang auch abgebrochen

werden kann. Die tatsächliche Kontextmarkierung wird erst erstellt, wenn der zweite Eckpunkt des Rahmens per Mausklick definiert wird.

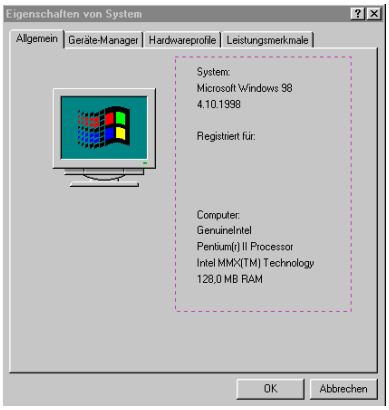



Abbildung 43: Vorgezeichnete Kontextmarkierung

Der Rahmen wird dann in der voreingestellten Farbe in der eingestellten Dauer angezeigt und mit den jeweiligen Ausmaßen an die anderen Konferenzteilnehmenden versendet.

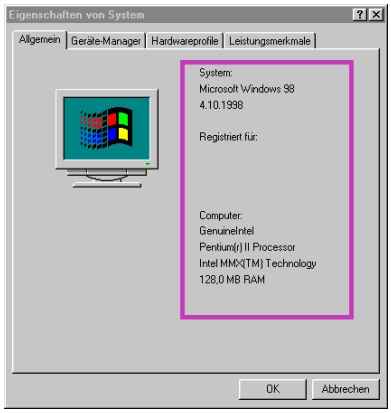



Abbildung 44: Flexibler Rahmen als Kontextmarkierung

3. **Fester Rahmen**

**Erstellung eines festen Rahmens**

Diese Kontextmarkierung wird durch die mittlere Maustaste aktiviert. Diese Taste kann auch durch die Kombination aus STRG-Taste und linker Maustaste emuliert werden. Die aktuelle Position des Maus-Cursors zum Zeitpunkt des Klicks definiert den Mittelpunkt des anzuzeigenden Rahmens in der voreingestellten Größe und Farbe. Der Rahmen wird für die Dauer der eingestellten Zeit angezeigt.

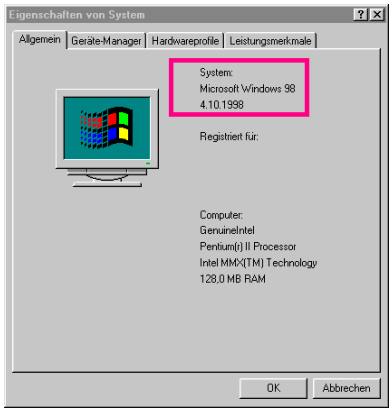



Abbildung 45: Fester Rahmen als Kontextmarkierung

**Pfeil als Kontextverweis**

4. **Bitmap als Verweis**

Diese Art der Kontextmarkierung wird mit der rechten Maustaste durchgeführt. Die aktuelle Position des Maus-Cursors zum Zeitpunkt des Klicks wird als Referenzpunkt (Anfaßpunkt) für die Darstellung des Pfeils benutzt, d. h., der Pfeil zeigt auf diese Position.

**Einfache Änderung während der Erstellung**

Der dargestellte Pfeil wurde im vorhergehenden Dialog ausgewählt. Um eine effiziente Dokumentation zu unterstützen, ist es möglich, während der Annotationserstellung diesen Pfeil temporär zu ändern. Durch die Kombination der Cursor-Tasten während der Betätigung der rechten Maustaste wird der entsprechende Pfeil angezeigt. Zum Beispiel verursachen die Cursor-Tasten oben und links die Darstellung eines Pfeils, der nach oben links verweist.

Das Bitmap wird an der aktuellen Cursor-Position für die eingestellte Dauer angezeigt und die notwendigen Informationen für die Anzeige bei den anderen TeilnehmerInnen versendet.

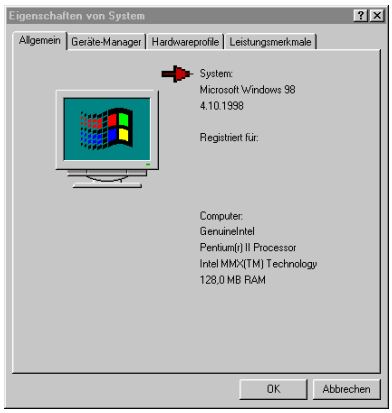



Abbildung 46: Pfeil als Kontextmarkierung

**Anzeige der Kontextmarkierung**

5. **Anzeigedauer der Annotation**

Die konkrete Kontextmarkierung wird in der eingestellten Anzeigedauer dargestellt. Diese Zeitspanne wird in Abhängigkeit der Einstellungen der Audio-Aufnahme durch unterschiedliche Anzeigen visualisiert. Die Aufnahme des verbalen Kommentars beginnt unmittelbar nach Aktivierung der Kontextmarkierung.

- Aktivierte Audio-Aufnahme
  Bei aktivierter Audio-Aufnahme zeigen das Statusfenster und der Maus-Cursor den Beginn der Aufnahme gleichzeitig an.

  **Laufende Audio-Aufnahme mit Anzeige der Kontextmarkierung**

  Nach Ablauf der Anzeigedauer der Kontextmarkierung ändert sich die Form des Maus-Cursors, um anzuzeigen, daß sich das System immer noch im Aufnahme-Status des Annotationsmodus befindet. Das Statusfenster ändert sich hingegen nicht.

  **Laufende Audio-Aufnahme nach Anzeige der Kontextmarkierung**

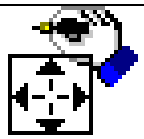
  Die Audio-Aufnahme wird durch erneutes Klicken einer beliebigen Maustaste gestoppt und das ErgoNet geht direkt in den Bereit-Zustand über.

- Deaktivierte Audio-Aufnahme
  Bei deaktivierter Audio-Aufnahme geht das ErgoNet-System nach Ablauf der Anzeigedauer direkt in den Wartezustand für die Erstellung einer neuen Annotation über.

  **Keine Audio-Aufnahme während der Anzeige der Kontextmarkierung**

### Übersicht der Maus-Cursor-Formen und Statusfenster

Die nachfolgende Tabelle gibt eine Übersicht der jeweiligen Maus-Cursorformen und Inhalte des Statusfensters zu den verschiedenen Annotationsstati.

| Statusbezeichnung | Maus-Cursor | Statusfenster |
|---|---|---|
| **Bereit** |  Hand mit Text „Start“ |  „Bleistift“-Grafik |
| **Flexibler Rahmen** |  Hand mit Stift und vier Pfeile in einem Rechteck |  „Screenshot mit Rahmen“-Grafik |
| **Fester Rahmen**[67] |  Hand mit Text „Start“ |  „Bleistift“-Grafik |

[67] Für diesen Status ist kein eigener Maus-Cursor und eigenes Statusfenster notwendig, da die Bearbeitung nur aus einem Schritt besteht.

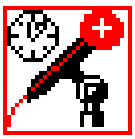
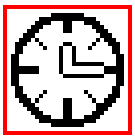
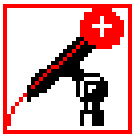
| Statusbezeichnung | Maus-Cursor | Statusfenster |
|---|---|---|
| **Bitmap als Verweis**[68] |  Hand mit Text „Start“ |  „Bleistift“-Grafik |
| **Aktivierte Audio-Aufnahme während der Kontextmarkierung** |  Mikrophon mit Uhr |  „Aktiviertes Mikrophon“-Grafik |
| **Deaktivierte Audio-Aufnahme während der Kontextmarkierung** |  Uhr |  „Bleistift“-Grafik |
| **Audio-Aufnahme nach Anzeige der Kontextmarkierung** |  Mikrophon |  „Aktiviertes Mikrophon“-Grafik |

Tabelle 6 Maus-Cursor und Statusfenster

### 5.5.7 Dokumentation

**Annotationen als Dokumentation**

Die während des Tests erstellten Annotationen stellen die Dokumentation des Tests und seiner Arbeitsaufgabe dar. Diese Annotationen, bestehend aus Screenshot mit Kontextmarkerierung und einer zugeordneten Kommentar-Datei, können mittels des Annotations-Viewers betrachtet und abgespielt werden, falls eine Audio-Datei aufgenommen wurde.

**Viewer bedient unterschiedliche Konferenzen bzw. Annotations-Sequenzen**

Damit dieser Viewer universell für unterschiedliche Konferenzen eingesetzt werden kann, besteht mit einem „Datei-Öffnen“-Dialog die Möglichkeit, Annotationen aus unterschiedlichen Konferenzen zu betrachten.

**Betrachtung im Rahmen einer Konferenz oder als eigenständige Applikation**

Die Betrachtung der Annotationen kann sowohl während einer Konferenz im Rahmen der Konferenzsteuerung als auch ohne eine bestehende Konferenz als eigenständige Applikation durchgeführt werden.

[68] Für diesen Status ist kein eigener Maus-Cursor und eigenes Statusfenster notwendig, da die Bearbeitung nur aus einem Schritt besteht.

Aktiviert wird der Viewer über den Menüpunkt „Annotationen/Betrachten...“ in der Konferenzsteuerung bzw. Auswahl des Eintrags „AnnotationViewer“ aus dem Startmenü.

**Verschiedene Aktivierungsmöglichkeiten**

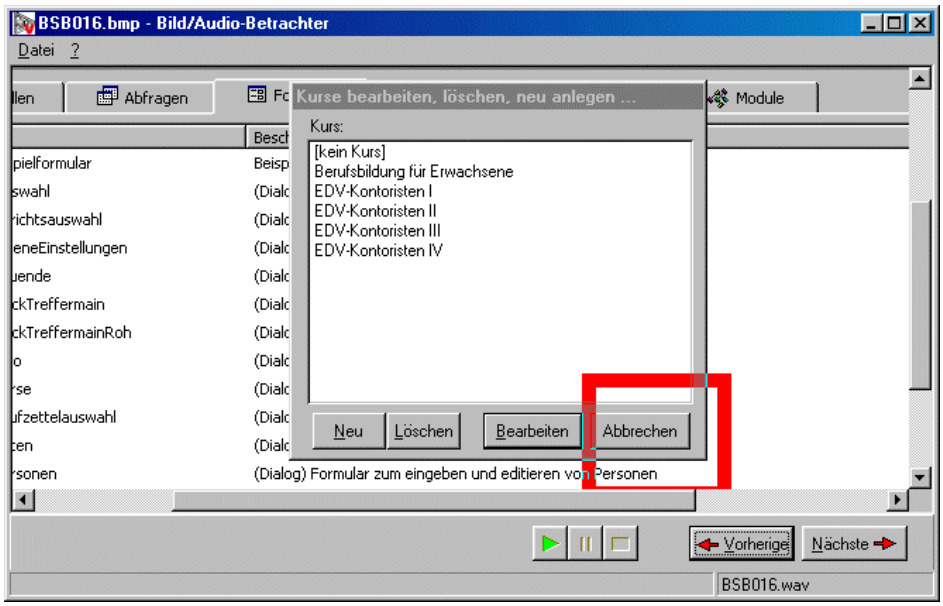



Abbildung 47: Annotations-Betrachter

Der Arbeitsbereich dieses Programms gliedert sich in drei Bereiche, wobei aber nur zwei optisch abgesetzt sind:

**Gliederung des Viewers in drei Bereiche**

1. **Grafikbereich**
   Im Grafikbereich werden die erstellten Screenshots mit den Kontextmarkierungen angezeigt, sofern diese mit erstellt wurden. Der Name des Screenshots wird mit in der Titelzeile der Applikation integriert.

   **Anzeigebereich des Screenshots**

   Die Größe dieses Bereichs paßt sich an die Größe der erstellten Screenshots bzw. die Größe des Fensters des Viewers an. Sollte ein Screenshot größer sein als dieser Bereich, so hat die BenutzerIn die Möglichkeit, per Scroll-Leisten den sichtbaren Bereich zu verschieben oder den sichtbaren Bereich durch Vergrößerung des Fensters des Viewers zu erweitern.

   **Individuelle Anpassung des Anzeigebereichs**

2. **Audio-Bedienung**
   Wurde bei der Annotationserstellung eine Kommentardatei aufgenommen bzw. dem aktuellen Screenshot zugeordnet, so kann diese Datei abgespielt werden. Der Name dieser Datei wird in der Statuszeile angezeigt.

   **Audio-Bedienung der Kommentar-Datei**

   Die Steuerung umfaßt den Start bzw. das Fortfahren, die Unterbrechung und das Stoppen der Wiedergabe, wobei die möglichen Aktionen durch eine farbliche Hinterlegung (grün für das Abspielen/Fortfahren, rot für Stoppen und gelb für die Unterbrechung) markiert sind.

   **Start/Fortfahren, Unterbrechen und Stoppen**

Abbildung 48: Audio-Steuerung

Wurde keine Audio-Datei aufgenommen, ist die Steuerung deaktiviert und in der Statuszeile vermerkt, daß keine Audio-Datei zugeordnet wurde.

3. **Annotations-Sequenz-Bedienung**

**Definition einer Sequenz**

Eine Sequenz von Annotationen definiert sich über den gleichen Prefix in einem gemeinsamen Verzeichnis. Der Index der Annotation, der gemeinsam mit dem Prefix als Dateiname verwendet wird, muß lükkenlos fortlaufend durchnumeriert sein.

**Bedienung einer Annotations-Sequenz**

Sofern die aktuelle Annotation ein Bestandteil einer Sequenz ist, kann mittels der Sequenz-Steuerung entschieden werden, ob zur nächsten Annotation oder zur vorherigen gesprungen werden soll.


Abbildung 49: Sequenz-Bedienung

Die Steuerung der Audio-Wiedergabe und Annotations-Sequenzen ist gemeinsam angeordnet, um so zwischen Anzeige- und Aktionsbereich zu unterscheiden.

## 5.6 Hilfsmodule

**Hilfsmodule von Windows integriert**

Die Ebene der Hilfsmodule haben wir nicht selbst implementiert, sondern die Mechanismen von der gewählten Plattform Windows9x übernommen. Folgende Funktionalitäten haben wir benutzt:

**Windows-Registry**

- **Registry-Datenbank**
  In der Registrierung von Windows werden zentral benutzerInnen-, anwendungs- und systemspezifische Konfigurationsdaten gespeichert. Diese Datenbank ist hierarchisch aufgebaut.

  Alle Konfigurationsdaten von ErgoNet werden in der Registrierungsdatenbank von Windows gespeichert. Ein Installationsprogramm garantiert, daß alle Schlüssel mit sinnvollen Werten vorbelegt sind.

**Windows-Messages**

- **Registrierung von Windows-Nachrichten**
  Um schmale Schnittstellen zwischen dem Basismodul und den einzelnen Funktionsmodulen zu garantieren, findet eine lose Kopplung über das Windows-Botschaften-System statt. Um sicherzustellen, daß Nachrichten nur innerhalb der Konferenzkomponenten ausgetauscht werden, werden die einzelnen Nachrichtenarten bei Windows registriert. Durch die Registrierung wird garantiert, daß systemweit eindeutige ID's für die Nachrichten vergeben werden.

**Datenstrukturen**

- **Datenstrukturen**
  Wir haben, soweit möglich, windowsinterne Datenstrukturen benutzt. Des weiteren haben wir die Mechanismen der Programmierplattform benutzt, um Listen, Tabellen etc. zu pflegen.

# 6 Entwicklungsverfahren und Implementierung von ErgoNet

**Ziele von Kapitel 6**

Wir werden uns in diesem Kapitel mit einigen Aspekten der Implementierung unseres Systems auseinandersetzen. Wir werden kurz die Prinzipien des prototypischen Systemspezifikationsprozesses und der Modularisierung im Softwareentwurf erläutern. Zu Beginn wollen wir in einem Exkurs unser persönliches Vorgehen bei der Erstellung unserer Diplomarbeit kritisch beleuchten.

## 6.1 Exkurs: Persönliches Vorgehen bei der Diplomarbeit

**CeBIT98**

Anfang Januar präsentierten uns Jürgen Friedrich und Uwe Haupt die Idee, im Rahmen einer Diplomarbeit einen funktionstüchtigen Prototypen von ErgoNet für die CeBIT98 19. bis 25.03.1998 zu entwickeln. Die Erfahrungen der CeBIT98 sollten uns helfen, das System zu vervollkommnen. Somit entwikkelten wir zunächst ein System, um anschließend in einem Analyseverfahren die theoretischen Fundamente auszuarbeiten. Diese Vorgehensweise ist in der Softwareentwicklung zwar nicht weit verbreitet, die Diskussion ist allerdings nicht ganz neu[69].

**Expert Online**

Die Idee des Systems wurde aus den Erfahrungen von Expert-Online[70] entwickelt. Expert-Online ist ein Beratungssystem, bei dem eine ExpertIn EntwicklerInnen berät. Das Ziel von Expert-Online ist dem von ErgoNet ähnlich: Die Erstellung software-ergonomisch guter Software. Die Diskussionen um Expert-Online auf der CeBIT97 ergaben, daß das Hinzuziehen der BenutzerInnen eine wesentliche Bereicherung des Systems wäre.

**Kooperation ISI und DMN**

ErgoNet ist als Kooperation zweier Teilinstitute des TZI – dem Institut für Software-Ergonomie und Informationsmanagement (ISI) und dem Bereich Digitale Medien und Netze (DMN) – angelegt. Die grundlegenden Eckpunkte von ErgoNet wurden von beiden Instituten wie folgt beschrieben[71]: „ErgoNet ist ein Beratungs- und Unterstützungsangebot, das aus einer Beratungs- und einer technischen Komponente besteht: Beratungsangebote der Ergonomie-Experten zur Formulierung von Prüfaufgaben, zur Auswertung der Online-Usability-Tests und zur ergonomischen Gestaltung der Software werden durch ein Internet-basiertes Konferenzsystem unterstützt. Dieses System vereint audiovisuelle Kommunikation mit Komponenten zur Anwendungskooperation und integriert alle Bestandteile in einer homogenen Benutzungsschnittstelle, die auch für Nichttechniker geeignet ist.“. Mit dieser Definition war das Fundament für unser System bereits gelegt.

**Technisches Herangehen**

Wir hatten zwei Monate Zeit, um einen funktionstüchtigen Prototypen für die Messe fertigzustellen. Ganz ohne Fundament, ohne konkrete Anforderungen kann jedoch auch kein „Basisprototyp“ entwickelt werden. So entwarfen wir ein Papier, in dem wir zunächst ein eher technisches Herangehen an das System zugrunde legten. Unser Ansatz in diesem Papier vom 13.01.1998 war aus den grundlegenden Eckpunkten von ErgoNet abgeleitet, nämlich die Kommunikationsbeziehungen von EntwicklerInnen, ExpertInnen und Benutzer-

[69] Vgl. die Diskussionen um Reverse-Engineering in REVERSE ENGINEERING.

[70] Die Flyer von Expert-Online und ErgoNet befinden sich im Anhang.

[71] Dieses Papier ist im Anhang einzusehen.

Innen zu analysieren. Aufgrund dieser Beziehungen und den technischen Rahmenbedingungen entwarfen wir eine Systemarchitektur von ErgoNet, die der jetzigen schon sehr nahekam.

**Inhaltliche Basis von ErgoNet**

In einem zweiten Schritt entwickelten wir in einem Papier vom 23.01.1998 eine inhaltliche Basis für ErgoNet. Dieses Herangehen bestand aus vier Schritten[72]:

1. Analyse der Interessenlagen von EntwicklerInnen, BenutzerInnen und ExpertInnen. Wir betrachteten hier zunächst den gesamten Entwicklungszyklus der Software von der Anforderungsdefinition bis zur Wartung des fertigen Systems.
2. Erläuterung der funktionalen Anforderungen des Systems. Dieses bearbeiteten wir vorerst allgemein. In einem zweiten Schritt zeigten wir tabellarisch einen Zusammenhang zwischen den Interessenlagen und den funktionalen Anforderungen der beteiligten Personen.
3. Aus den Interessenlagen, den funktionalen Anforderungen und den anderen Informationen, die wir über ErgoNet hatten, entwickelten wir Fragestellungen, die für die weitere Arbeit beantwortet werden müssen.
4. Wir entwarfen danach erste technische Überlegungen zur Umsetzung der erarbeiteten Interessenlagen von EntwicklerInnen, BenutzerInnen und ExpertInnen.

**Beide Ansätze waren wichtig**

Aus diesen beiden Ansätzen entstand der erste Prototyp unseres Systems. Sowohl der technische als auch der inhaltliche Ansatz waren für uns wichtig. Der technische Ansatz spannte einen Rahmen, in dem wir uns bewegen wollten. Der inhaltliche Ansatz führte zu konkreten Anforderungen, die wir in unserem ersten Prototypen umgesetzt haben.

**Netmeeting, Basis der Konferenzsteuerung**

Im Vordergrund stand die Umsetzung der Basisanforderungen: Konferenzsteuerung und Annotation. Frühzeitig geriet Netmeeting in den Mittelpunkt unseres Interesses. Netmeeting verfügte über ein ActiveX, das über verschiedene Programmierplattformen eingebunden werden konnte. Unsere ersten Versuche machten wir mit MS-Visual Basic, weitere kleinere Prototypen folgten in Borland-Delphi 2.0. Inhaltlich war die Idee der homogenen Benutzungsschnittstelle vorrangig. Leider war die ActiveX-Schnittstelle sehr fehlerträchtig, so daß wir die Arbeit mit diesem Ansatz einstellen mußten. Das ActiveX war eine Einbahnstraße in der Kommunikation mit Netmeeting. Alle Events von Netmeeting wurden an das Netmeeting-User-Interface gesendet, so daß wir sie nicht eigenständig auswerten konnten.

**Netmeeting COM-Schnittstelle**

Der zweite Versuch war die Nutzung der COM-Schnittstelle (vgl. Kapitel 6.3) von Netmeeting. Mit Hilfe dieser Schnittstelle waren wir in der Lage, das Netmeeting-User-Interface vollständig durch unsere eigene Konferenzsteuerung zu ersetzen.

**Globale Maus- und Tastatursteuerung**

Ein weiteres Aufgabengebiet war die globale Maus- und Tastatursteuerung von Windows. Wir benötigten die verschiedenen Systemevents, bevor Win-

[72]Dieses Papier ist im Anhang einzusehen.

dows sie verarbeitete und sie damit für uns nicht mehr verfügbar waren. Mit Hilfe von Newsgroups und diversen API-Büchern gelang es uns schließlich, die Mausereignisse abzufangen, bevor Windows sie an die Applikationen weiterleitete – das Kernstück der Annotation war beendet (vgl. Kapitel 6.3).

**Problem Audio**

Die anderen Teile (Audio- und Videoübertragung) schienen trivial, waren doch gängige Tools im Internet weit verbreitet. Zu einem Problem wurde dann aber die Realisierung der Audioübertragung mit Hilfe eines geeigneten Tools. Zunächst waren diese Tools nicht styleguidekonform (TCL/TK). Über eine Fernsteuerung mit Hilfe eines lokalen Sockets überwanden wir diesen Mangel, weil die Tools so nicht mehr sichtbar sein mußten (vgl. Kapitel 5.5.4).

**Tag der offenen Tür beim TZI**

Nach der CeBIT98 sind wir über eine weitere Verfeinerung unseres Prototypen schrittweise zu dem System gelangt, wie es sich jetzt darstellt. Wir begannen mit der theoretischen Aufarbeitung und bereiteten uns auf den zweiten Test beim Tag der offenen Tür des TZI vor. Durch weitere Verfeinerungen durch Diskussion mit den ExpertInnen und Ad-Hoc Inspektionen wurde das System so gestaltet, wie es in Kapitel 5 vorgestellt wurde.

## 6.2 Entwicklungsverfahren

**Erstellung eines Basisprototypen**

Wie im vorigen Kapitel bereits verdeutlicht, wurde der erste Basisprototyp mit Hilfe sehr rudimentärer Anforderungen – quasi Hypothesen – erstellt. Dieser Basisprototyp wurde dann schrittweise verfeinert und verändert. Diese Veränderungen sind beispielsweise bei der Benutzungsoberfläche der Konferenzsteuerung zu beobachten:

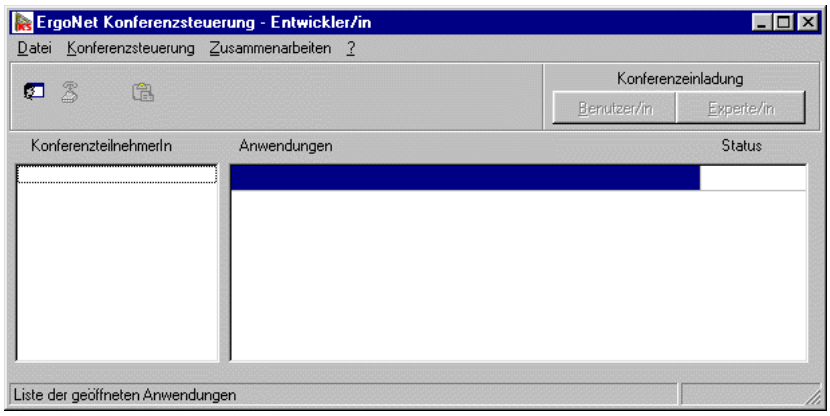



Abbildung 50: Konferenzsteuerung erster Basisprototyp

Der obige Basisprototyp wurde im Verlauf der Iterationen verändert, so daß er sein derzeitiges Aussehen bekam:

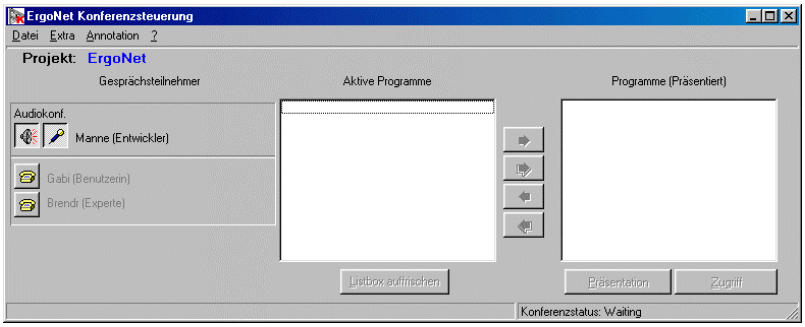



Abbildung 51: Konferenzsteuerung nach Spezifikation

### 6.2.1 Prototyporientierter Systemspezifikationsprozeß

**Prototyporientierter Systemspezifikationsprozeß**

Nach der Präsentation des Basisprototypen auf der CeBIT98 haben wir „prototypisch-orientiert" das System bis zur Präsentation beim Tag der offenen Tür des TZI weiterentwickelt. Bei dem prototyporientierten Systemspezifikationsprozeß bildet „die Ermittlung, die Beschreibung und die Evaluation der Anforderungen" (vgl. POMBERGER & BLASCHEK 1996 S. 44) das Rückgrat.

**Basis-Anforderungen**

Ausgangspunkt einer solchen Entwicklungsstrategie ist die Ermittlung von (Basis-) Anforderungen und die Erstellung eines (Basis-) Prototyps. Diesen Prozeß haben wir im Januar 1998 durchgeführt, im wesentlichen in der Diskussion mit Ergonomie-Experten in der Universität Bremen. Anhand dieser groben Anforderungen wurde der erste Prototyp mit den wichtigsten funktionalen und datenbezogenen Eigenschaften des geplanten Systems modelliert (vgl. POMBERGER & BLASCHEK 1996 S. 45).

**Iterativer Prozeß**

In einem iterativen Prozeß wurden folgende Schritte durchgeführt:

- Detaillierte Beschreibung der Systemfunktion.
- Änderung und Erweiterung des Prototyps.
- Experimente mit dem Prototyp.
- Analyse von Systemfunktion und Benutzerschnittstelle.

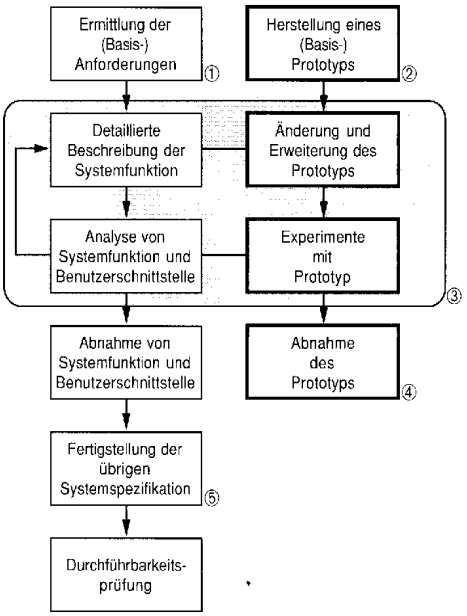



Abbildung 52: Die prototypische Systemspezifikation: (POMBERGER & BLASCHEK 1996)

Die Zusammenfassung aller Anforderungen, die in den verschiedenen Iterationen gesammelt wurden, sind in Kapitel 4 beschrieben.

### 6.2.2 Modularisierung

**Eigenschaften von Modulen**

Nachdem die Anforderungen an ein System formuliert wurden (vgl. Kapitel 4), haben wir das System spezifiziert bzw. entworfen. Der erste Entwurfsschritt ist, die Gesamtaufgabe in Teilaufgaben zu zerlegen, also das Problem zu modularisieren. Ein Modul soll dabei folgende Eigenschaften haben (vgl. POMBERGER & BLASCHEK 1996 S. 53):

- Ein Modul ist eine Zusammenfassung von Operationen und Daten zur Realisierung einer abgeschlossenen Aufgabe.
- Die Kommunikation eines Moduls mit der Außenwelt darf nur über eine eindeutig spezifizierte Schnittstelle erfolgen.
- Zur Integration eines Moduls in ein Programmsystem darf keine Kenntnis seines inneren Arbeitens erforderlich sein.
- Die Korrektheit eines Moduls muß ohne Kenntnis seiner Einbettung in ein Programmsystem nachprüfbar sein.

**Feinentwurf**

Die Modularisierung fand bei uns in Funktionseinheiten statt, wie sie in Kapitel 5 beschrieben wurden. Jedes Modul muß in einem Feinentwurf weiter zergliedert werden (vgl. POMBERGER & BLASCHEK 1996 S.58ff):

- Entwurf der abstrakten Datentypen und Objekte
- Entwurf der Datenstrukturen
- Entwurf der Schnittstellen
- Entwurf der Benutzungsoberfläche.

**Module zu autarken Programmen entwickeln**

Um eine optimale Arbeitsteilung zu erreichen, wurden die einzelnen Programmteile parallel erarbeitet. Daher wurden die einzelnen Module zunächst als autarke Programme entwickelt, die wir später durch geeignete Schnittstellen zusammenführten. Der positive Begleiteffekt ist, daß die verschiedenen Module durch sehr schmale Schnittstellen gekoppelt werden.

### 6.2.3 Evaluierung durch Ad-hoc-Inspektionen und Praxistests

**Evaluation**

Jeder Entwicklungsschritt wurde mit einer Evaluierung durch Ad-hoc-Inspektionen von Ergonomie-Experten abgeschlossen. Neben den Praxistests, die wir in Kapitel 7 beschreiben, fanden mehrere Tests im SELab[73] statt. Hier wurden sowohl die Technik als auch die verschiedenen inhaltlichen Aspekte der Software getestet und evaluiert.

Im Laufe der Entwicklung wurden mehrere Praxistests durchgeführt:

- CeBIT98.
- Tag der offenen Tür des TZI.
- Test der Annotation in Zusammenarbeit mit der Bremer Schule Bürgerweide (BSB).
- Test mit verteilten ISDN-Teilnehmern mit dem Kohne Ingenieurbüro GmbH.

Die Ergebnisse der einzelnen Tests sind jeweils in die weitere Entwicklung des Systems eingeflossen (vgl. Kapitel 7).

## 6.3 Eingesetzte Techniken

**Einblick in technische Zusammenhänge**

Im folgenden werden wir einige Techniken beschreiben, die wir im Verlauf der Implementierung eingesetzt haben. Über die inhaltlichen Umstände hinaus wollen wir damit einen Einblick in die inneren technischen Zusammenhänge unseres Systems ermöglichen.

### WM_COPYDATA

**Gekapselte Adreßräume pro Applikation**

Die Entscheidung, die einzelnen Module eigenständig zu entwickeln, führt unter Windows9x dazu, daß die Module gekapselte Datenbereiche besitzen. Es ist nicht möglich, daß ein anderer Prozeß von außen auf diese Datenbereiche zugreifen kann.

**Verschiedene IPC-Methoden unter Windows**

Um jedoch einen Datenaustausch zwischen den Modulen zu realisieren, muß ein Mechanismus zur Interprozeßkommunikation eingesetzt werden. Windows bietet u. a. die folgenden an[74]:

**Shared Memory**

- ***Shared Memory* in *Dynamik Link Labraries* (DLL)**
  In einer DLL wird ein Speicherbereich mehreren Prozessen zugänglich gemacht (vgl. PETZOLD 1996 S. 1081ff).

[73] Software-Ergonomie-Labor.

[74] Die Zwischenablage wird in diesem Zusammenhang nicht als anwendbarer Mechanismus diskutiert, da hierbei nicht kontrolliert werden kann, wer wann welche Daten in diesem gemeinsamen Bereich verändert hat.

- ***Object Linking and Embedding (OLE)***
  OLE
  Diese Technik erlaubt es, mittels einer Reihe von Spezifikationen Softwarekomponenten zu erstellen, die miteinander kommunizieren können (vgl. PETZOLD 1996 S. 1103ff).
- ***Dynamic Data Exchange (DDE)***
  DDE
  Dieser Austausch von Daten basiert auf dem Austausch von Meldungen, der in einer Client/Server-Beziehung stattfindet (vgl. PETZOLD 1996 S. 981ff).
- ***WM_COPYDATA***
  WM_COPYDATA
  Mit diesem Nachrichtentyp ist es möglich, einen bestimmten strukturierten Adreßbereich einem anderen Prozeß zugänglich zu machen. Dieser Bereich wird kopiert und mittels Zeigeroperationen anderen Prozessen zugänglich (*read-only*) gemacht (vgl. DELPHI 2 1996, PROSISE 1997).

Diese verschiedenen Methoden haben wir für unsere Einsatzzwecke geprüft, und entschieden, daß das Kopieren von strukturierten Adreßbereichen unsere Anforderungen optimal unterstützt.

**Indizierte WM_COPYDATA-Nachrichten**

Da das Datenaufkommen, das zwischen den einzelnen Modulen ausgetauscht werden muß, immer die gleiche Struktur aufweist, wird diese allen Modulen bekanntgemacht (globale Strukturdefinition). Durch Versenden einer indizierten *WM_COPYDATA*-Nachricht können die Empfängerapplikationen entscheiden, für welchen Zweck (z. B. Anzeigen einer Kontextmarkierung) die Daten benutzt werden sollen.

### COM-Schnittstelle

**Deklarative Schnittstelle**

Ab Delphi 3.0 unterstützt Borland auch die COM-Technik von Windows. Die COM-Schnittstelle ist eine rein deklarative Schnittstelle. Obwohl in diesem Zusammenhang von Objekten gesprochen wird und es auch einen Vererbungsmechanismus von COM-Objekten gibt, bestehen deutliche Unterschiede zu anderen Klassenbibliotheken:

1. **COM - abstrakte Objekte**
   Im COM-Modell werden Schnittstellen (*interfaces)* bereitgestellt. Die verschiedenen Funktionalitäten einer Applikation (z. B. Netmeeting) werden auf ein Objektmodell abstrakt umgesetzt. Die Deklaration der Schnittstellen hat nichts mit der tatsächlichen Implementierung zu tun.
2. **Verschiedene interfaces für gleiche Objekte**
   Die *interfaces* werden an ein COM-Objekt gebunden. Da die *interfaces* rein deklarativ sind, können auch verschiedenartige *interfaces* an das gleiche COM-Objekt gebunden werden, um so verschiedene Funktionen des Objekts zu nutzen. Der Vorteil eines solchen Mechanismus ist, daß gleiche Methodenaufrufe für verschiedene Funktionaliäten bereitgestellt werden können.
3. **Keine richtige Objekt-Semantik**
   Aus den Gründen von 2. gibt es zwar Vererbungsmechanismen der *interfaces,* aber keinen Polymorphismus somit kann in diesem Fall im strengen Sinne nicht von Objekten gesprochen werden.

**Verschiedene Arten von interfaces**

Im COM-Modell werden zwei Arten von *interfaces* unterschieden: *Interfaces*, die auf ein bereits bestehendes COM-Objekt verweisen (wir nennen diese Schnittstellen im folgenden Basisinterfaces), und *interfaces,* die noch keine Implemen-

tierung haben (vgl. RUBENKING 1997). Diese *interfaces* verweisen oft auf Objekte, die eine Ereignisbehandlung durchführen, und werden im folgenden Notifyobjekte genannt. Die Implementierung der Notifyobjekte muß in dem jeweiligen Programm erfolgen, das die Funktionalität benutzt, die die Basisinterfaces zur Verfügung stellen.

**IDL-Skripte und Typenbibliotheken**

Das Netmeeting SDK stellt ein Reihe von Basisinterfaces und *interfaces* für die Notifyobjekte zur Verfügung. Die *interfaces* liegen in IDL (*interface description language*) vor und müssen zunächst delphispezifisch aufgearbeitet werden. Dies geschieht unter Delphi mit Hilfe von Typenbibliotheken, die wir aus dem IDL-Skript entwickelt haben.

**Unterschiedliche COM-Standards**

Nachdem wir die Typenbibliothek erstellt hatten, implementierten wir die Notifyobjekte. Wir haben die Notifyobjekte objektorientiert entwickelt. Eine Basisklasse enthält alle Funktionen, die notwendig sind, ein Notifyobjekt mit einem COM-Objekt zu verbinden (*connect, disconnect*). Alle abgeleiteten Objekten implementieren dann die spezifischen Funktionen der Notifyobjekte. Die Implementierung der Notifyobjekte gestaltete sich sehr komplex, da Microsoft und Borland unterschiedliche COM-Standards implementiert haben.

### Audio-API

**Zwei Soundkarten, da jeweils nur ein Aufnahmekanal verfügbar ist**

Mit dem Betriebssystem Windows und dem Einsatz der weitverbreiteten Soundblaster-kompatiblen Soundkarten besteht die Notwendigkeit, zwei Soundkarten im System einzusetzen. Die Soundkarten bieten jeweils nur einen Aufnahmekanal, der in der jeweiligen Anwendung gekapselt wird.

**Kapseln der Audio-Daten in der Audio-Übertragungskomponente**

ErgoNet setzt in der Audio-Übertragungskomponente eine externe Applikation[75] ein, die diese Audio-Daten intern verwaltet und verarbeitet, ohne daß die Möglichkeit besteht, diese Daten im ErgoNet-System für weitere Zwecke weiterzuverwenden. Dies ist aber notwendig, um beispielsweise auf der einen Seite Audio-Daten an die Konferenz-Teilnehmenden zu versenden und auf der anderen Seite diese Daten für die Kommentar-Datei der Annotation bei der ExpertIn zu speichern.

**Bestimmung der Anzahl der Soundkarten und Zuordnung der Standard-Soundkarte für rat**

ErgoNet prüft während des Starts der Annotations-Komponente, ob das Rechnersystem über mindestens zwei Soundkarten verfügt. Ist dies nicht der Fall, so wird die Audio-Aufnahme des Annotations-Tools deaktiviert. Ist jedoch eine zweite Karte eingebaut und verfügbar, so ist es unter Windows95 erforderlich, diese zweite Soundkarte explizit anzusteuern, während die erste standardmäßig[76] von *rat* belegt ist.

**Drei Ebenen der MM-API**

Die Windows-Multimedia-API bietet mehrere Schnittstellen zur Ansteuerung eingebauter Soundkarten an. Diese Schnittstellen gliedern sich in drei Ebenen, die sich aus der Hardwarenähe der jeweiligen Funktionen ableiten:

1. **High-Level (*MCIWnd*)**
   Die High-Level Schnittstelle ist eine Bibliothek mit Funktionen und Makros. Mit dieser Schnittstelle werden nicht die Geräte, sondern die

[75] Z. B. *rat*, *vat*, CuSee-Me oder IBM BambaPhone.

[76] Unter der Systemsteuerung/Multimedia von Windows95 ist es möglich einzustellen, welche der eingebauten Soundkarten die Standard-Soundkarte für Aufnahme bzw. Wiedergabe sein soll.

Dateien (*wave, avi*), angesprochen. Mit Hilfe dieser Bibliothek können Audio- und Videodateien schnell aufgenommen und abgespielt werden (vgl. MULTIMEDIA API 1997 S. 957ff).

2. **Mid-Level (MCO-API)**
   Diese Schnittstelle stellt eine geräteunabhängige Schnittstelle zur Verfügung. Die Befehle werden mit Hilfe von Befehlsnachrichten übermittelt und arbeiten nicht nur auf Dateien (wie *MCIWnd*), sondern auch auf Geräteebene (vgl. MULTIMEDIA API 1997 S. 863ff).[77]

3. **Low-Level**
   Für Anwendungen, bei denen eine absolute Kontrolle von Multimedia-Geräten und ein direkter Zugriff auf die Wavedateien erforderlich ist, stellt Windows mehrere Low-Level-Schnittstellen zur Verfügung. Die Low-Level-Waveform-Audiofunktionen beispielsweise erlauben eine genaue Steuerung beim Abspielen und Aufzeichnen von Audiostreamen (vgl. MULTIMEDIA API 1997 S. 672ff).

Für unsere Zwecke kam nur die Low-Level-Schnittstelle in Betracht, weil nur mit ihr die einzelnen Geräte (Soundkarte 1 und Soundkarte 2) separat angesprochen werden konnten. Die anderen beiden Schnittstellen arbeiteten nur auf das Standarddevice, also auf die in der Windows-Multimedia eingestellte Standardsoundkarte.

Die grundlegende Vorgehensweise beim Aufzeichnen und Abspielen von Waveform-Audio vollzieht sich in folgenden Schritten:

1. **Öffnen des gewünschten Geräts**
   Mit dieser Funktionalität kann bereits geprüft werden, ob alle gewünschten Geräte verfügbar sind.

2. **Allozierung von Speicher**
   Es werden ein oder mehrere Datenpuffer alloziert. Über mehrere Speicher ist es möglich, größere Datenmengen zu bearbeiten. Es empfiehlt sich, doppelt zu puffern, so daß ein Aufnahmegerät einen Datenpufferblock verarbeiten und die Anwendung den anderen speichern kann. So entstehen keine Lücken im Ton.

3. **Vorbereiten der Puffer**
   Bevor die Puffer Daten aufnehmen können, müssen sie vorbereitet werden. Dazu dienen spezielle Funktionen, mit denen unter anderem die Aufnahmequalität festgelegt werden kann.

4. **Füllen der Puffer, abspeichern**
   Die Puffer werden von den Geräten gefüllt. Nach dem Füllen enthalten die Puffer die rohen Audiodaten. Nach dem Abspeichern müssen noch spezifische Headerinformationen vorgeschaltet werden, damit Windows die Dateien als Wave-Daten verwalten kann. Die Ermittlung

[77] Eine solche Nachrichtenfolge ist z. B.:

```
open cdaudio
set cdaudio time format tmsf
play cdaudio from 1 to 6
close cdaudio
```

der genauen Informationen des Headers erwies sich als schwierig, weil mehrere WAV-Spezifikationen unter Windows existieren (vgl. BORN 1990).

5. **Freigabe der Ressourcen**
   Zum Schluß müssen die Ressourcen freigegeben werden. Dies gilt für den allozierten Speicher wie für das verwendete Gerät.

Die oben genannten Schritte entfallen unter Windows98, weil es mehrere Soundkarten unterstützt. In der Windows98-Multimedia-Konfiguration kann als Standardsoundkarte „nächste verfügbare" eingestellt werden. Dadurch wird ein Konflikt verschiedener Soundkarten vermieden. Zur Entwicklungszeit von ErgoNet stand Windows98 leider noch nicht zur Verfügung.

### Windows-Nachrichtenfilter (Hook)

**Eingriff in die normale Arbeitsweise von Windows**

Die Erstellung der Kontextmarkierungen in der Annotations-Komponente basiert auf Funktionen, die sehr tief im Windows-System verankert sind. Aus diesem Grund ist es notwendig, in die „normale" Arbeitsweise von Windows einzugreifen und diese so zu verändern, daß die notwendigen Daten verfügbar und veränderbar sind.

**Umleitung der Mausaktionen und -bewegungen**

Die Markierungen, die standardmäßig auf der verteilten Applikation erstellt werden, müssen in den entsprechenden Anzeigebereich integriert werden. Für eine einfache und effiziente Erstellung ist es erforderlich, die Mausaktionen und -bewegungen in diesem Anzeigebereich an ErgoNet bzw. die Annotationskomponente umzuleiten.

**Message-Hooks**

Diese Funktionalität bieten die sogenannten Hook-Funktionen unter Windows. Mit deren Einsatz können bestimmte Nachrichten, die im Windows-System verteilt werden, abgefangen und verarbeitet werden (vgl. WIN32API 1996 S. 191ff). Zu diesen Nachrichtentypen gehören z. B. Tastatur-, Shell- und Maus-Nachrichten.

- In der Annotationskomponente haben wir den Typ der Maus-Hook-Funktion (WH_MOUSE) eingesetzt (vgl. WIN32API 1996 S. 191), die folgende Nachrichten filtert und an die Annotations-Komponente umleitet. Grundsätzlich werden dabei die Nachrichten draufhin unterteilt, ob sie sich auf den Innenbereich (Client-Bereich)[78] oder Steuerungsbereich (Non-Client-Bereich)[79] eines Fensters beziehen (vgl. PETZOLD 1996 S. 338ff):

**Maus-Nachrichten im Client-Bereich**

**Maus-Nachrichten im Client-Bereich**

| Nachricht | Bedeutung |
|---|---|
| WM_LBUTTONDOWN | Linke Maustaste gedrückt |
| WM_MBUTTONDOWN | Mittlere Maustaste gedrückt |
| WM_RBUTTONDOWN | Rechte Maustaste gedrückt |
| WM_MOUSEMOVE | Maus bewegt |

[78] Hierzu gehört der umrahmte Teil des Fensters, der nicht von Titelleiste, Menü oder Bildlaufleisten belegt wird (vgl. CUBER & WENZEL 1994 S. 33f).

[79] Hierzu gehört der Bereich, der nicht im Client-Bereich liegt.

**Maus-Nachrichten im Non-Client-Bereich**

Maus-Nachrichten im Non-Client-Bereich

| Nachricht | Bedeutung |
|---|---|
| WM_NCLBUTTONDOWN | Linke Maustaste gedrückt |
| WM_NCMBUTTONDOWN | Mittlere Maustaste gedrückt |
| WM_NCRBUTTONDOWN | Rechte Maustaste gedrückt |
| WM_NCMOUSEMOVE | Maus bewegt |

Alle Nachrichten enthalten als zusätzliche Parameter die jeweiligen Positionskoordinaten der Maus.

Hook-Funktion als Filter

Die Hook-Funktion, die selbst in einer DLL gekapselt ist, wird von der Annotations-Komponente durch den Wechsel in den Annotationsmodus installiert. Von diesem Moment an werden die oben genannten Nachrichten nicht mehr vom Windows-System, sondern durch die Annotations-Komponente verarbeitet. Durch die Beendung des Annotationsmodus wird die Hook-Funktion wieder entfernt und die Nachrichten wieder an das Windows-System weitergeleitet.

In der DLL wird nach Erhalt einer Nachricht entsprechend reagiert und mittels des WM_COPYDATA-Mechanismus die Annotations-Komponente darüber informiert, welche weiteren Aktionen folgen müssen:

1. **WM_LBUTTONDOWN/WM_NCLBUTTONDOWN ohne STRG-Taste**

   Linke Maustaste ohne STRG-Taste

   Die Annotationskomponente beginnt mit der Erstellung eines flexiblen Rahmens durch Vorzeichnen eines gestrichelten Rahmens entsprechend den Maus-Bewegungen (WM_MOUSEMOVE). Falls sich das System bereits in dieser Einstellung befand, wird die eigentliche Kontextmarkierung erstellt.

   Befindet sich das System im Aufnahmemodus einer Kommentar-Datei, wird diese Aufnahme gestoppt.

2. **Bei WM_MBUTTONDOWN/WM_NCMBUTTONDOWN oder WM_LBUTTONDOWN/ WM_NCLBUTTONDOWN und STRG-Taste**

   Mittlere Maustaste oder Linke Maustaste mit gehaltener STRG-Taste

   Die Annotationskomponente erzeugt einen festen Rahmen in der eingestellten Größe und Farbe, sofern sich das System nicht in der Erstellung eines flexiblen Rahmens oder in der Aufnahme einer Kommentar-Datei befindet. In diesen Fällen wird die Erstellung abgebrochen bzw. die Aufnahme beendet.

3. **WM_RBUTTONDOWN/ WM_NCRBUTTONDOWN ohne Cursor-Taste**

   Rechte Maustaste mit/ohne Cursor-Taste

   An der Position des Maus-Cursors wird das aktuell ausgewählte Bitmap eingeblendet. Sollte beim Maus-Klick eine der Cursor-Tasten betätigt sein, wird das entsprechende Bitmap angezeigt.

Wenn sich das System in der Erstellung eines flexiblen Rahmens oder im Aufnahmemodus der Kommentar-Datei befindet, wird die Einstellung abgebrochen bzw. gestoppt.

**Überprüfung des Tastatur-Status**

Zusätzlich wird in der DLL zur Hook-Funktion der aktuelle Status der Tastatur geprüft (vgl. WIN32API S. 406f), um entscheiden zu können, ob die mittlere Maustaste emuliert oder in der Annotation ein anderer Pfeil angezeigt werden soll.

Durch die Kombination dieser verschiedenen Funktionen können, die notwendigen Daten für die Kontextmarkierung ermittelt werden. Diese Daten werden durch WM_COPYDATA-Nachrichten an die Annotations-Komponente gesendet, die anschließend für die Anzeige der Markierungen verwendet wird. Hierfür ist die Windows-Schnittstelle der Grafik-API zuständig.

### Grafik-API

**Anzeige der Kontextmarkierungen mittels Grafik-API**

Zum Anzeigen der Kontextmarkierungen besteht die Notwendigkeit zur Nutzung der untersten, also hardware-nächsten Schicht der Windows-Schnittstelle, da diese Markierungen in Fenstern angezeigt werden sollen, die nicht vom ErgoNet-System erstellt und verwaltet werden können.

**Kein Zugriff auf den Zeichnungsmechanismus der Fenster**

Unter Windows sind keine Mechanismen und Funktionen vorgesehen, den Anzeigebereich anderer Fenster kontrolliert zu manipulieren, da jedes Fenster seinen eigenen (Neu-)Zeichnungsmechanismus kapselt, der von außen nicht zugänglich ist.

**Ermittlung des jeweiligen Fensters bzw. der ID**

Für das Zeichnen der Kontextmarkierungen entsprechend den Nachrichten, die von der Maus-Hook-DLL gesendet wurden, muß zunächst ermittelt werden, in welchem Fenster die erhaltenen Koordinaten liegen (vgl. WIN32API 1996 S. 84f).

**Unterbinden des Neuzeichnens**

Dies ist notwendig, da das Neuzeichnen dieses Fensters für die Anzeigedauer der Kontextmarkierung unterbunden werden muß. Andernfalls würde diese Markierung nur „aufflackern“, da der Zeichnungsmechanismus des Fensters beim nächsten Neuzeichnen diese Markierung wieder entfernen würde. Wird an der entsprechenden Position kein Fenster gefunden, wird der dahinterliegende Desktop als Fenster eingesetzt und entsprechend behandelt.

**Verschiedene Stiftarten**

Anschließend werden die Rahmen der Markierung mittels der Zeichenfunktionen der Grafik-API gezeichnet, wobei die Stiftarten u. a. folgende Einstellungen unterscheiden (vgl. PETZOLD 1996 S. 152):

| Linientyp | Bedeutung |
|---|---|
| PS_SOLID | Durchgezogen |
| PS_DOT | Gepunktet |
| PS_DASH | Gestrichelt |

**Farbe der Markierungen**

Die Farbe der Linien wird ebenfalls über die Attribute des Zeichenstifts eingestellt, wobei die tatsächlich gezeichnete Farbe davon abweicht. Die jeweilige Farbe ergibt sich aus der eingestellten Linienfarbe und der Hintergrundfarbe jedes Pixels der Linie. So kann es vorkommen, daß die sichtbare Linie aus

mehreren Farben besteht, wenn sich der Hintergrund aus verschiedenen zusammensetzt. Die gezeichnete Linie ist also sichtbar, ohne daß der jeweilige Hintergrund verdeckt wird.

**Rasteroperation-Codes(ROP)**

Diese Farbenzusammenstellung wird in der entsprechenden Windows-Schnittstelle Rasteroperation (vgl. PETZOLD 1996 S. 155ff, WIN32API 1996 S. 841 und S. 878) genannt. Es werden verschiedene Codierungen angeboten, wobei wir uns für R2_NOTXORPEN entschieden haben, die eine inverse Kombination aus Linien- und Hintergrundfarbe verursacht.

**Restaurierung durch wiederholtes Zeichnen**

Zusätzlich bietet diese Codierung die Möglichkeit, durch ein einfaches erneutes Zeichnen der Markierung den Original-Zustand des Fensters wiederherzustellen, ohne daß sich dieser Zustand „gemerkt" werden mußte.

**Kopieren der Pfeile in den Anzeigebereich**

Die Pfeile als Kontextverweis werden anders erstellt als die Rahmen als Kontextmarkierung. Das liegt daran, daß die Pfeile eigentlich nicht gezeichnet werden, sondern als fertige Bitmaps vorliegen und nur in den Anzeigebereich des Fensters kopiert werden (vgl. PETZOLD 1996 S. 227ff, WIN32API 1996 S. 601ff).

**Restaurierung durch Speicherung**

Da diese Kopien nicht mit den oben genannten Rasteroperationen kombiniert werden können, ist es notwendig, den von der Kopie veränderten Bereich zu speichern, damit er nach Ablauf der Anzeigedauer des Kontextverweises restauriert werden kann.

# 7 Erprobung und Präsentation

Im Laufe der Entwicklung des Systems fanden mehrere Präsentationen statt. Alle Erprobungen hatten unterschiedliche Zielsetzungen und Abläufe. Im nachfolgenden beschreiben wir die einzelnen Tests in ihrer zeitlichen Reihenfolge. In jedem Kapitel werden wir neben den Zielen den Ablauf der Erprobung sowie Schlußfolgerungen beschreiben.

## 7.1 CeBIT98

**Erste öffentliche Vorstellung des Systems**

Die erste öffentliche Präsentation von ErgoNet fand im Rahmen der CeBIT98 vom 19. - 25.03.1998 in Hannover statt. Auf einem eigenen Stand innerhalb des Norddeutschen Gemeinschaftsstandes wurde auf einem PC das System installiert und präsentiert[80]. Der Gemeinschaftsstand befand sich in der Halle 9 (Forschungshalle) der CeBIT98, die einen Teil der aktuellen Forschungsgebiete zusammenfaßte.

**Situationsbeschreibung**

Auf der Messe haben wir die Arbeit der EntwicklerIn dargestellt, während die BenutzerIn und ExpertIn sich in der Universität Bremen im Software-Ergonomie-Labor aufhielten. Über eine gemietete Standleitung (500Kbit/s), die mit anderen Ständen des TZI geteilt wurde, haben die Konferenzteilnehmenden ihre Kommunikations-Verbindung aufgebaut.

**Ablauf**

Zwischen den Teilnehmenden wurde ein Szenario durchgespielt, das die Anwendungsmöglichkeiten des Systems den interessierten BesucherInnen der Messe vermittelte. Das Szenario umfaßte ein fiktives System, mit dem mittels HTML-Seiten Computer im WorldWideWeb ausgesucht und bestellt werden können. Dieses System wurde auf software-ergonomische Fragestellungen untersucht. Die ExpertIn beobachtete gemeinsam mit der EntwicklerIn die Arbeit der BenutzerIn. Die ExpertIn erstellte parallel dazu Annotationen und schickte sie anschließend an die EntwicklerIn, damit das System modifiziert werden kann.

**Ziele der Präsentation auf der CeBIT98**

### 7.1.1 Ziele

- **Präsentation des Systemprototyps auf der Messe durch den Ablauf eines Szenarios**
  **Reale Bedingungen**
  Durch das „Durchspielen“ des Szenarios unter realen Bedingungen, d. h. den Umständen, denen die Teilnehmenden unterliegen (z. B. örtliche Trennung, *wide area*-Netzwerkverbindung), mußte das System seine Leistungsfähigkeit unter Beweis stellen.

- **Technische Erprobung des Systems zwischen Messe und Uni**
  **Erste technische Erprobung**
  Da das System zum Zeitpunkt des Messebeginns noch nicht unter realen Bedingungen[81] getestet werden konnte, bedeutete diese Präsentation auch die erste vollständige technische Erprobung.

[80] Standaufsicht während der Messe hatten neben uns: Peter Ansorge, Guido Frick, Uwe Haupt.

[81] Mit realen Bedingungen ist hier die Entfernung und der Einsatz des Internets gemeint.

- **Inhaltliche Darstellung des Themengebietes Software-Ergonomie auf der Messe**
  Über die Vorstellung des Systemprototyps hinaus wurde das Themengebiet Software-Ergonomie und die Bedeutung für den Software-Entwicklungsprozeß durch uns den interessierten BesucherInnen erläutert.

  **Vorstellung der Software-Ergonomie**

- **Anregungen durch das Fachpublikum der Messe für das gedachte oder weitere Einsatzgebiete des Systems**
  Ein weiteres Ziel der Messeaktivitäten war es für uns, von den MessebesucherInnen Anregungen für den Einsatz des Systems zu erhalten. Vor allem waren wir an Anstößen interessiert, die das System als Unterstützung bei ergonomischen Sachfragen sehen. Andere Richtungen traten in den Hintergrund, waren aber trotzdem aufschlußreich.

  **Anregungungssuche**

- **Repräsentation der Universität Bremen bzw. des TZI**
  Der Stand auf der Messe fand unter der Schirmherrschaft der Universität Bremen und des TZI statt, so daß wir mit unserer Mitarbeit diese Institutionen nach außen vertraten und Informationen über die vielfältigen Aktivitäten verbreiten konnten.

  **Vertreter des TZI**

### 7.1.2 Ablaufbeschreibung

Um eine umfassende Übersicht aller Aktivitäten zu erhalten, unterteilen wir die Beschreibung des Ablaufs in die Vorbereitungsphase, den Ablauf und eine Nachbereitungsphase.

**Vorbereitungsphase**

Die Vorbereitungsphase, die ca. vier Wochen vor Messebeginn einsetzte, diente dazu, alle notwendigen Vorbereitungen für die Messe zu analysieren und abzuschließen. Hierzu gehörten vor allem:

**Beginn der Vorbereitung**

- **Abschluß der zentralen Programmiertätigkeiten**
  Diese Arbeiten konnten rechtzeitig abgeschlossen werden, so daß alle notwendigen Änderungen abgearbeitet werden konnten (z. B. geänderte technische Anforderungen, auftretende Programmfehler), die sich im Laufe der Vorbereitung ergeben sollten.

  **Beendigung der Programmierung**

- **Beschaffung und Vorbereitung der notwendigen Hardware**
  Die Hardware[82], die für die Präsentation auf der Messe erforderlich war, mußte noch beschafft, d. h. neu gekauft werden, da die vorhandenen Geräte technisch überholt und damit nicht einsatzfähig waren.

  **Hardwarebeschaffung**

- **Erarbeiten einer Beschreibung des Präsentationsrahmens auf der Messe**
  Das Szenario, das auf der Messe präsentiert werden sollte, wurde in Zusammenarbeit mit Peter Ansorge erstellt. Unser Vorschlag, einen Arbeitsablauf bei der Abrechnung von Hafengebühren in Zusammen-

  **Szenarioerstellung**

[82] Hierzu gehören drei adäquat ausgestattete Computersysteme, Headsets, Lautsprecher und Videokameras.

arbeit mit dem Hansestadt Bremischen Hafenamt (HBH)[83] als Aufgabe heranzuziehen, wurde als zu komplex verworfen. So erstellte Peter Ansorge das „Heckenbusch-System“ [84] auf HTML-Basis, das von uns auf der Messe präsentiert wurde. Die Rolle der BenutzerIn und der ExpertIn wurde während der Messe von verschiedenen Personen übernommen.

**Zusammenarbeit mit dem DMN**

- **Koordination der Aktivitäten mit Vertretern des DMN**
  Die gesamt Arbeit war als Kooperation des ISI und DMN geplant und wurde dementsprechend durchgeführt. Während der Erstellung von ErgoNet konzentrierten wir uns auf die Umsetzung der Hauptkomponenten des Systems, während das DMN in Person von Matthias Bock die Aufgabe übernahm, die Audio- bzw. Videoübertragung durch externe Werkzeuge zu realisieren. Er prüfte verschiedene Alternativen, erstellte eigene Versionen und entschied gemeinsam mit uns, das Werkzeug *rat* einzusetzen. Die notwendigen technischen Voraussetzungen (z. B. Realisierung der Multicast-Anbindung) wurden in Zusammenarbeit mit dem DMN umgesetzt.

**Vorbereitungen auf der Messe und in SELab**

- **Aufbau des Messestandes und Abschluß aller notwendigen Vorbereitungen im SELab**
  Am Tag vor Messebeginn wurden sowohl im SELab der Universität Bremen als auch auf der Messe in Hannover alle notwendigen Aufbauten und weiteren Vorbereitungen abgeschlossen. Im SELab war es anschließend möglich, einen Testdurchlauf des Szenarios[85] zu starten, um den Teilnehmenden den konkreten Ablauf näherzubringen.

### Ablauf

**Audio-Übertragung als Problem**

Während der Messe setzte sich die Zusammenarbeit mit den Vertretern des DMN fort, indem sie weiter an den Problemen der Audio-Übertragung[86] und Multicast-Realisierung arbeiteten. Parallel wurden im SELab andere Alternativen[87] auf ihre Einsetzbarkeit hin untersucht.

Da sich diese Probleme im weiteren Verlauf nicht beheben ließen, konzentrierten wir uns bei der Präsentation auf die lokale Funktionalität des Systems. Die Konferenzsituation und der damit verbundene Einsatz des Systems wurde

[83] Vormals Hansestadt Bremisches Amt (HBA).

[84] Die entsprechenden Dokumente befinden sich im Anhang auf der CD.

[85] Hierbei wirkten Peter Ansorge, Guido Frick und Joachim Hinrichs mit, die während der Messe auch die entsprechenden Rollen übernahmen.

[86] Im Gegensatz zur Video-Übertragung konnte keine Full-Duplex-Verbindung zwischen der Messe und der Universität hergestellt werden, die den grundsätzlichen qualitativen Anforderungen gerecht wurde. Es war nicht möglich, diesen Umstand während der Messe zu beheben bzw. die Ursachen hierfür zu beschreiben.
Diskussionen und einen Erfahrungsaustausch gab es mit MessebegleiterInnen des Deutschen Forschungs-Netzes (DFN), die eine ähnliche Ausstattung und Präsentation im Hinblick auf Audio- und Videoübertragung hatten. Sie schilderten vergleichbare Probleme, die sie aber durch entsprechend angepaßte Inhalte der Präsentation verschleierten.

[87] Beispielsweise das Zusammenschalten von zwei Mikrophonen an einer Soundkarte, um eine Dreierkonferenz über eine Strecke zu realisieren.

verbal geschildert, während die Erstellung und die Einsatzmöglichkeiten der Annotation anhand des erstellten Szenarios verdeutlicht wurden.

**Publikum der CeBIT98**

Über das System hinaus wurden Fragestellungen, die durch software-ergonomische Aspekte im Entwicklungsprozeß eines Programms aufgeworfen werden, diskutiert und erläutert. Zentrale Gegenstände waren dabei der Stellenwert, der Zeitpunkt und die Integrationsmöglichkeiten software-ergonomischer Mechanismen im Software-Entwicklungsprozeß. Da nicht nur deutschsprachiges, sondern auch internationales Publikum anwesend und interessiert war, untermauerten ihre Äußerungen unsere Annahmen, daß software-ergonomische Aspekte und Fragestellungen weit verbreitet auf einer gemeinsamen Basis diskutiert werden können.

Die Rückmeldungen während der Messe waren vielfältig und informativ. Die nachfolgende Liste faßt die Reaktionen zusammen:

- **Weitere Einsatzmöglichkeiten**

  **ErgoNet als HelpDesk**

  Die verbale Schilderung des Einsatzes von ErgoNet zur Durchführung eines Ad-Hoc Usability-Tests reichte aus, den BesucherInnen zu verdeutlichen, was das System leisten kann. Viele merkten hierzu an, daß eine Anwendung der beschriebenen Funktionalitäten im Rahmen eines HelpDesk-Systems innovativ sei. Durch die Zusammenführung der EntwicklerInnen und ExpertInnen könnte den BenutzerInnen eine optimale Möglichkeit geboten werden, bei konkreten Fragestellungen bzw. Problemen mit bestimmten Systemen kompetente Unterstützung zu erhalten.

  Weiter wurde angeregt, dieses System als Diskussionsforum für EntwicklerInnen und ExpertInnen zu nutzen, um alternative Designentwürfe zu bearbeiten. Durch den *shared application*-Mechanismus sind die Voraussetzungen geschaffen, gemeinsam an einem Rapid-Prototyp-System zu arbeiten und verschiedene Entwürfe zu gestalten.

- **Notwendigkeit der Software-Ergonomie und eines Systems zur Unterstützung dieser Fragestellungen**

  **Notwendigkeit für ErgoNet**

  Die allgemeinen Fragestellungen der Software-Ergonomie wurden ausführlich, und wir halten fest, daß die Notwendigkeit eines unterstützenden Systems besteht.

- **Keine Notwendigkeit eines fertigen Systems**

  Während der Messe genügte es, das System mit seiner lokalen Funktionalität der Annotations-Erstellung und -Darstellung zu präsentieren. Den interessierten BesucherInnen genügte eine verbale Beschreibung der Konferenz-Funktionalität, um die Einsatzmöglichkeiten des Systems zu erkennen.

### Nachbereitungsphase

Die Erfahrungen der Messe, daß die Audio-Übertragung ein grundsätzliches Problem darstellt, führte zu der Erkenntnis, daß in diesem Bereich weitere Werkzeuge gesucht und gefunden werden müssen. Ohne die Audio-Übertragung, ein wesentlichen Bestandteil des Systems, kann keine effektive Unterstützung bei Usability-Problemen gewährt werden.

### 7.1.3 Ergebnisse

- Die allgemein positive Resonanz bedeutete eine Bestätigung unserer bisherigen Arbeit und zeigte die Notwendigkeit eines derartigen Systems auf.
- Die weiteren Einsatzmöglichkeiten des Systems bzw. seiner Funktionalitäten sollten weitergehend analysiert und entsprechend umgesetzt werden.
- Da keine konkreten Anregungen für weitere Entwicklungen gewonnen werden konnten, haben wir entschieden, daß der funktionale Rahmen des Systems vollständig ist und die Aspekte der Benutzungsschnittstelle bei der weiteren Arbeit nun in den Vordergrund rücken.
- Das Problem Audioübertragung muß weiter analysiert werden, oder es muß auf andere technische Lösungen (z. B. Integration von Telefon in den PC) ausgewichen werden. In der verfügbaren Literatur wird davon ausgegangen, daß zeitkritische Übertragungen mit der derzeitigen Struktur des Internets ohne Reservierungs- und Bandbreitenmechanismen noch nicht ausgereift sind: „Da IP bisher weder echtzeitfähig ist noch Dienstqualität garantiert, treten gelegentlich Fehler bei der Sprachübertragung auf." (ANDERSEN & SCHMIDT 1998, S. 268). Für die nahe Zukunft wird dieses allerdings als Standard angekündigt: „Auch wenn man dem MBone und den zugehörigen Tools den experimentellen Charakter noch deutlich anmerkt, läßt sich bereits erkennen, daß Multicast das Potential hat, das Internet radikal zu verändern." (ANDERSEN & SCHMIDT 1998, S. 268)[88].

### 7.1.4 Schlußfolgerungen/Bemerkungen

**Anstrengungen nach der CeBIT98**

Aus dem Verlauf der Messe und der Analyse der erarbeiteten Ergebnisse haben wir folgende Schlußfolgerungen für weitere Iterationsschritte in der Systementwicklung gezogen:

- Es fand eine Überarbeitung der Benutzungsschnittstelle im Rahmen einer Evaluation mit einem Ergonomie-Experten (Guido Frick) statt.
- Um den Annotations-Mechanismus zu optimieren, haben wir eine Verfeinerung der Funktionalität (flexible Rahmen, Verweise, geänderte Steuerung) durchgeführt.

## 7.2 Tag der offenen Tür TZI

**Zweiter Praxistest**

Nach der CeBIT98 führten wir einen zweiten Praxistest beim Tag der offenen Tür des TZI am 26.06.1998 durch. Bei dieser Erprobung haben wir uns auf die Durchführung eines Szenarios konzentriert, um das System in seiner Gesamtheit zu testen. Alle involvierten Personen agierten in einem lokalen Netz.

[88] Weitere Informationen zu „Telefonie in IP-Netzen" finden sich bei PC PROFESSIONAL 1998 und IMMLER 1999, wo zusammenfassend gesagt wird: „Die Telefonier-Server heute sind nur eine moderne Alternative zu CB-Funk." (IMMLER 1999 S. 137).

Aus diesem Grund waren technische Probleme wie auf der CeBIT98 theoretisch auszuschließen.

### 7.2.1 Ziele

**Ziele des zweiten Tests**

Bei dieser zweiten technischen Erprobung hatten wir folgende Ziele:

- Erprobung des Gesamtsystems mit drei Rechnern, Audio- und Videoübertragung, *shared application* und Annotation.
- Durchführung eines Szenarios mit einer kleinen Arbeitsaufgabe zur Überprüfung des Gesamtablaufs eines Ad-hoc Usability-Tests.
- Diskussion mit Interessierten über das Gesamtsystem mit dem Ziel, Verbesserungen vorzuschlagen.

Insgesamt diente uns der Tag der offenen Tür als zweite technische Erprobung. Mit der Durchführung in einem lokalen, multicastfähigen Netz (100Mbit/s Ethernet) stellten wir sicher, daß die gewählten Audio- und Videotools einwandfrei funktionieren und daß wir uns damit auf die Inszenierung der Konferenz und der Annotation konzentrieren konnten.

### 7.2.2 Ablaufbeschreibung

#### Vorbereitung

**Überarbeitung nach CeBIT98**

Die Vorbereitungsphase für den Tag der offenen Tür begann unmittelbar nach der CeBIT98. Neben der Überarbeitung der Benutzungsoberfläche und der Suche nach geeigneten Audio- und Videotools achteten wir vor allem darauf, das System zu stabilisieren.

**Erprobungen im SELab**

Als Vorbereitung fanden mehrere Erprobungen im SELab statt. Das SELab wurde nach der Messe mit neuen Rechnern ausgestattet. Wir konzentrierten uns ganz auf die Erprobung in einem lokalen Netz, weil dies als Konfiguration im TZI geplant war.

#### Durchführung

**Technische Probleme**

Die Erprobung fand in den Räumen des TZI statt. Beim Aufbau unserer drei Rechner stellten wir fest, daß ein Rechner nicht bootete. Wir stellten zwei Rechner auf, den der ExpertIn und den der BenutzerIn. Konzeptionell wollten wir die Rechner der ExpertIn und den der EntwicklerIn nebeneinander stellen, um deutlich zu machen, wie eine EntwicklerIn die Annotationen sieht, die die ExpertIn durchführt. Durch den Ausfall eines Rechners konnten wir unsere Überlegungen nicht umsetzen.

**Schwerpunkt waren Vorträge**

Schwerpunkt des Tags der offenen Tür beim TZI waren Vorträge zu diversen informatischen Themen. Die Vortragspausen nutzten die Teilnehmenden um verschiedene Exponate zu sehen, die auf vier Etagen verteilt waren. Somit gab es Zeiten, in denen viele die Exponate besichtigten und andere, in denen kaum jemand anwesend war.

### 7.2.3 Ergebnisse

**Wenig Interesse**

Während des gesamten Tages gab es verschiedene Reaktionen, die sich hauptsächlich auf den Einsatzzweck des System zur Unterstützung eines Usability-Tests bezogen:

- Die Idee des Ad-hoc Usability-Tests wurde positiv aufgenommen. Das Prinzip der Vereinfachung ohne Verzicht auf große Qualitätseinbußen wurde hervorgehoben, wobei gleichzeitig darauf hingewiesen wurde, daß bei sicherheitsrelevanten Systemen auf umfassende Tests nicht verzichtet werden darf. Unser System zur Unterstützung von Ad-hoc Usability-Tests wurde als zweckmäßig eingestuft, da wir die gesamte Konzeption des Systems auf ein zusätzliches Werkzeug gelegt haben, ohne herkömmliche Methoden ersetzen zu wollen.
- Analog zur CeBIT98 wurden wir darauf hingewiesen, daß das System auch als Support-System eingesetzt werden kann. Die Funktionalität des *application sharing* würde hierbei als entfernte Steuerung bei Problemen eingesetzt werden.

**Ergebnisse des Tags der offenen Tür**

Die Ergebnisse dieses Tests fassen wir folgendermaßen zusammen:

- Im lokalen Netz hatten wir sowohl mit der Video- als auch mit der Audioübertragung keine Probleme, so daß die MBone-Tools zum Einsatz kommen konnten.
- Durch die Überarbeitung nach der CeBIT98 sind fehlerhafte Funktionen sowie software-ergonomische Mängel minimiert worden. Das System erwies sich als stabil.
- Die Besucher fanden sowohl die Idee als auch deren Umsetzung eine sinnvolle Ergänzung gängiger Qualitätssicherungsmethoden.
- Wie auf der CeBIT98 wurden auch bei dieser Veranstaltung Vorschläge zur Erweiterung des Systems in Richtung HelpDesk diskutiert.

### 7.2.4 Schlußfolgerungen

**Positives Resümee**

Aufgrund der Resonanz auf unser System zogen nach dem Tag der offen Tür wie schon bei der CeBIT98 ein positives Resümee. Die Erprobung hat uns gezeigt, daß sich das System durch unsere Änderungen technisch ausgereifter zeigen konnte als in vorherigen Präsentationen.

## 7.3 Praxistest Bremer Schule Bürgerweide

**Inhalte des Tests**

Der Praxistest in Zusammenarbeit mit der Bremer Schule Bürgerweide (BSB) hatte die Prüfung der Arbeitsweise und Einsatzfähigkeit der Annotationskomponente als zentralen Aspekt. Wir verzichteten auf den hierfür unnötigen Overhead der Konferenzsituation und führten den Test ohne direkte Kommunikation zwischen EntwicklerIn und ExpertIn durch. Da die Annotationen hauptsächlich von der ExpertIn für die EntwicklerIn erstellt werden, konnte auf eine Integration von BenutzerInnen verzichtet werden. Innerhalb dieses Praxistests wurde keine Arbeitsaufgabe durchgeführt, sondern ein reines *clean up* der Benutzungsschnittstelle in Form einer Ad-hoc Inspektion.

**Zwei Phasen dieses Tests**

Der Test selbst bestand aus zwei völlig getrennten Phasen, so daß sich die Analyse des Praxistests ebenfalls in zwei Abschnitte gliedert. Die folgende Tabelle verdeutlicht diese Gegenüberstellung:

| Test | Analyse |
|---|---|
| Erstellung der Annotationen auf ExpertIn-Seite | Arbeitsweise und Integrationsfähigkeit der Annotationserstellung in den Arbeitsprozeß der ExpertIn |
| Betrachtung der Annotationen und Hilfestellung für die Weiterentwicklung auf EntwicklerIn-Seite | Aussagefähigkeit der Annotationen für konkrete Hilfestellungen bei der Weiterentwicklung |

Tabelle 7: BSB-Test und Analyse

Das zu testende System ist als Unikat für die Bremer Schule Bürgerweide (BSB) auf der Entwicklungsplattform MS-Access97 entwickelt worden. Das System wurde als Client-Server-Applikation konzipiert und umgesetzt. Sowohl Clients als auch der Datenbankserver sind in MS-Access97 implementiert.

**BSB führt Arbeitsamtmaßnahmen durch**

Die Bremer Schule Bürgerweide führt verschiedene Bildungsmaßnahmen für das Arbeitsamt Bremen durch. Ziel ist es, daß die Teilnehmenden einen Arbeitsplatz finden. Um dieses Ziel zu erreichen, werden Betriebspraktika vermittelt sowie ein besonderer Service für die Bewerbungsberatung und -durchführung angeboten.

**Ziele des zu testenden Systems**

Das zu testende System soll es der BSB erleichtern, beim Arbeitsamt oder in Zeitungen angebotene Stellen Teilnehmenden der Maßnahmen zuzuordnen. Die Teilnahme an dem Service ist freiwillig. Über einen Fragebogen wird das Profil der Arbeitssuchenden ermittelt und in einer Datenbank erfaßt. In einem zweiten Schritt werden Stellen erfaßt und halb- oder vollautomatisch mit den Profilen der Arbeitssuchenden verglichen. Im Anschluß an eine manuelle Nacharbeit werden die Arbeitssuchenden ausgegeben, die dann gezielt auf eine Stelle angesprochen werden können.

**Ziele auf ExpertIn-Seite**

### 7.3.1 Ziele des Tests auf ExpertIn-Seite

- Die technische Arbeitsweise des Annotations-Systems muß überprüft und evtl. Fehlfunktionen müssen erkannt werden.
- Alle notwendigen Arbeitsschritte sollen erkannt und ihre Integration in die Arbeitsweise der ExpertIn analysiert werden.
- Der Einsatz der verschiedenen Markierungsarten und deren jeweilige Wirkungsweise müssen untersucht werden.
- Die Nachbereitungsphase der Dokumentation (z. B. Zusammenstellung der Annotationen und Verschicken) muß beurteilt werden.

### 7.3.2 Ablaufbeschreibung des Tests auf ExpertIn-Seite

**Vorbereitungsphase**

In der Vorbereitungsphase erhielten wir das komplette System von der EntwicklerIn per E-Mail zugeschickt und installierten es auf unserem Computer[89].

**Vorgeschaltetes Gespräch**

In anschließenden Gesprächen erläuterte die EntwicklerIn den Einsatzzweck des Systems und das bisherige manuelle Vorgehen der späteren BenutzerInnen. Gemeinsam wurde eine Arbeitsaufgabe gestaltet, die die Arbeitsabläufe verdeutlichen sollte.[90] Anhand dieser Arbeitsaufgabe sollte die ExpertIn die einzelnen Dialoge untersuchen, wobei der Schwerpunkt der Analyse aber auf der „reinen" Oberflächengestaltung lag.

**Ablauf**

**Aufteilung in verschiedene Dialoge**

Der Ablauf des Praxistests bestand in der Betrachtung der Arbeitsaufgabe und den dazugehörigen Dialogen. Zu jedem Dialog wurden Annotationen erstellt, die die jeweiligen Probleme verdeutlichten und Lösungsvorschläge enthielten.

Im Laufe des Tests stellte sich heraus, daß es sinnvoll ist, diesen Test ein weiteres Mal mit den Änderungen der ersten Iteration durchzuführen.

**Nachbereitung**

Die Nachbereitung beschränkte sich auf die telefonische Besprechung der Annotationen, die für die EntwicklerIn unklar geblieben sind.

### 7.3.3 Ergebnisse des Tests auf ExpertIn-Seite

- Die Annotations-Komponente, die hier als eigenständige Applikation[91] zum Einsatz kam, ist ohne funktionale Fehler.
- Wir haben erkannt, daß die Qualifizierung der ExpertIn auf den Gebieten der Software-Ergonomie und des Arbeitskontextes sehr hoch sein muß. Nur mit einer derartigen Kombination ist es möglich, aussagekräftige Annotationen zu erstellen, die es der EntwicklerIn erlauben, diese Anmerkungen zu verstehen und umzusetzen.
- Die Erstellung von unterstützenden Annotationen ist relativ komplex, da auf der einen Seite die verbalen Anmerkungen kurz, aber aussagekräftig sein müssen. Auf der anderen Seite müssen die Kontextmarkierungen so gewählt werden, daß sich die Problembereiche eindeutig identifizieren lassen.
- Die Steuerung der Annotations-Erstellung ist dem Arbeitsablauf der ExpertIn angepaßt und gut integriert. Wir haben eine Zwei-Schirm-Lösung getestet, wobei die Steuerung der Annotation auf dem einen Bildschirm und der eigentliche Testgegenstand auf dem anderen zu

[89] Die Installationsroutine war nicht Gegenstand des Tests.

[90] Diese Aufgabe verdeutlichte letztlich nur den Zusammenhang zwischen den einzelnen Arbeitsschritten, ohne sich auf die eigentliche Arbeitsweise der BenutzerInnen zu konzentrieren.

[91] Aufgrund der Modularität unseres Systems konnte der fast identische Quelltext hierfür eingesetzt werden.

sehen waren. Diese Lösung erwies sich für die ExpertIn als ausgesprochen nützlich, da somit alle Informationen für die Annotationserstellung von denen des Testgegenstands getrennt waren. Gleichzeitig erkannten wir allerdings, daß diese Technik bei der Erstellung des Systems und bei der Implementierung der Testgegenstände berücksichtigt werden muß, da andernfalls Probleme auftauchen[92].

- Eine Sequenz-Erstellung bzw. die nachträgliche Strukturierung der Annotation wird im derzeitigen Prototypen noch nicht optimal gefördert und muß effektiver durch das System unterstützt werden.
- Die Funktionalität des Annotations-Betrachters als Kontrollinstrument für die ExpertIn ist ausreichend und als Grundlage für eine weitere Entwicklung zum „Organisator" der Dokumentation geeignet.

### 7.3.4 Schlußfolgerungen/Bemerkungen zum Test auf ExpertIn-Seite

**Einsatzanalyse der Annotationskomponente**

Zusammenfassend haben wir erkannt, daß die Funktionalität der Annotationskomponente für den Einsatzbereich der EntwicklerInnen angemessen und für die Erstellung aussagekräftiger Hilfestellungen ausreichend ist. Die Komponente ist einsetzbar, ohne daß gravierende Fehler auftraten, so daß in diesem Bereich keine weiteren Arbeiten notwendig sind. Diese Aussage läßt sich auch auf den Bereich der Steuerung übertragen, die sich optimal in die notwendigen Arbeitsschritte der EntwicklerIn integrierte.

Hierbei erschien uns die weitere Betrachtung einer Zwei-Schirm-Lösung folgerichtig (vgl. MODZELEWSKI 1998), die allerdings durch uns als Entwickler des ErgoNet-Systems nicht kontrollierbare Auswirkungen auf die Testgegenstände hätte.

**Weitere Anforderungen für eine Erweiterung des Systems**

Weitere Anforderungen haben sich für die Strukturierung der Annotationen ergeben, da der Viewer nur reine Anzeige-Funktionaliät bereitstellt. Eine parallele oder nachträgliche Strukturierung der Annotationen ist nicht möglich, und die Sequenzerstellung ist nur rudimentär umgesetzt worden. Diese Funktionalität ist jedoch in der Anforderungsermittlung für weitere Entwicklungen des Systems vorgesehen.

### 7.3.5 Ziele des Tests auf EntwicklerIn-Seite

**Ziele auf EntwicklerIn-Seite**

Das wesentliche Ziel auf der EntwicklerIn-Seite war es festzustellen, ob eine EntwicklerIn die Annotationen der ExpertIn problemlos verstehen und umsetzen kann. Konkret waren die Ziele:

- Welche Fehlerarten werden mit Hilfe des Systems gefunden und dokumentiert? Insbesondere war für uns interessant: Können ergonomische Katastrophen mit Hilfe von Annotation dargestellt werden?
- Wir wollten eine Verteilung verschiedener Fehlerarten ermitteln.

[92] Hierzu gehört beispielsweise, daß Kontext-Menüs auf einem anderen Bildschirm dargestellt wurden als auf dem der Applikationen. Manche Programme lassen es auch prinzipiell nicht zu, auf zwei Bildschirmen parallel zu arbeiten.

- Wir wollten das *impact ratio* ermitteln, also das Verhältnis von gefundenen zu korrigierten Fehlern.

### 7.3.6 Ablaufbeschreibung des Tests auf EntwicklerIn-Seite

**Zwei Testdurchführungen**

Der Test wurde in zwei Iterationen innerhalb zweier Wochen durchgeführt. Die EntwicklerIn bekam die Annotationen per E-Mail und entschied eigenständig, welche Anmerkungen umgesetzt werden sollten und welche nicht. Nachdem das System überarbeitet worden war, führten wir den zweiten Iterationsschritt durch.

### 7.3.7 Ergebnisse des Tests auf EntwicklerIn-Seite

Insgesamt ermittelten wir beim ersten Test 43 und beim zweiten Test 28 Fehler. Wir fanden beim zweiten Mal deshalb noch so viele Fehler, weil inzwischen neue Funktionen hinzugefügt wurden. Im folgenden stellen wir tabellarisch die Ergebnisse dar.

**Zusammenfassung der gefundenen Fehler**

Beim Analysieren der Fehler entwickelten wir eine Fehlertaxonomie, um die Fehler zusammenfassen zu können:

- **Konsistenzfehler**
  Dies sind Fehler, die sich in unterschiedlichen Überschriften, Bezeichnern oder Schaltern (Abbrechen vs. Abbruch) darstellen.
- **Styleguideverletzungen**
  Getestet wurde in der Windows-Umgebung. In dieser Fehlerklasse werden Mängel erfaßt, die eindeutig gegen den Windows-Styleguide verstoßen, z. B. keine Abgrauung von Feldern, die nicht editiert werden können.
- **Kosmetische Fehler**
  Zu dieser Fehlerklasse gehören Fehler, die in den bisher genannten Fehlerklassen nicht erfaßt sind, aber das Erscheinungsbild trübten. Häufige Fehler waren Bezeichner, die nicht genau untereinanderstanden, oder Gruppierungen, die teilweise um ein Pixel versetzt waren, aber auch Rechtschreibfehler.
- **Beeinträchtigung der Selbstbeschreibungsfähigkeit**
  Hier faßten wir Fehler zusammen, bei denen eine bestimmte Beschreibung einer Aktion nicht eindeutig war, beispielsweise Schalterbezeichnungen, die irreführen.
- **Ergonomische Katastrophen**
  Zu dieser Fehlerklasse gehören alle Fehler, nach deren Auftreten mit dem Programm nicht mehr weitergearbeitet werden kann, z. B. Tabelleneinträge, die nicht mehr editiert oder gelöscht werden können, Programmabstürze, die nicht über ein Fehlermanagement abgefangen werden können.
- **Falsche Fehlermeldungen /fehlendes Feedback**
  Unter diese Kategorie fallen Fehlermeldungen, die nicht auf den tatsächlichen Fehler hinweisen, oder fehlende Hinweise bei langen Operationen.

Die folgende Tabelle verdeutlicht die Fehler, die in den verschiedenen Fehlerklassen gefunden wurden. Wir teilen die Fehler außerdem in drei Schweregrade ein (vgl. Kapitel 2.6.2) von leicht (1) bis schwer (3):

| Kl. | Fehlerklasse | Test 1 | | Test 2 | |
|---|---|---|---|---|---|
| 1 | Konsistenzfehler | 11 | 26 % | 6 | 21 % |
| 1 | Styleguideverletzungen | 7 | 17 % | 2 | 8 % |
| 1 | Kosmetische Fehler | 13 | 30 % | 6 | 21 % |
| 2 | Beeintr. der Selbstbeschreibungsfähigkeit | 4 | 9 % | 3 | 11 % |
| 2 | Falsche Fehlermeld. / fehlendes Feedback | 2 | 5 % | 4 | 14 % |
| 3 | Ergonomische Katastrophen | 6 | 14 % | 7 | 25 % |
| | Gesamt gefundene Fehler | 43 | 100 % | 28 | 100 % |

Tabelle 8: Gefundene Fehler beim Praxistest mit der BSB

Das *impact ratio* lag bei beiden Tests bei über 90 (93 und 96). Wir können folgende Schlußfolgerungen aus dem Test ziehen:

- Es werden verschiedene Fehlerklassen gefunden. Im wesentlichen findet durch eine solche Methode ein *clean up* der Benutzungsoberfläche statt, d. h., es werden überwiegend leichte Fehler gefunden (jeweils über 50 %, im ersten Test sogar mehr als zwei Drittel).
- Bei der Verteilung der Fehler gibt es eine Tendenz: Bei der zweiten Inspektion verschiebt sich der Schweregrad nach oben, weil die kosmetischen und Konsistenzfehler meist in der ersten Iteration gefunden werden (von 1,41 nach 1,75).
- Bei unseren Tests war das *impact ratio* sehr hoch, weil im wesentlichen leichte Fehler gefunden wurden, die schnell behoben werden konnten.

## 7.4 Erprobung in Zusammenarbeit mit dem Kohne Ingenieurbüro GmbH

**Erprobung mit allgemein verfügbaren Ressourcen**

Die letzte Erprobung unseres Systems fand in Zusammenarbeit mit dem Kohne Ingenieurbüro GmbH statt. Die bisherigen Tests fanden in großen Netzen statt (MBone, Ethernet). Ausgelegt ist ErgoNet jedoch für Netzwerkressourcen, die allgemein verfügbar sind (vgl. Kapitel 4).

### 7.4.1 Ziele des Tests

**Getestet wurde der Konferenzaufbau**

Im Mittelpunkt dieses Tests stand nicht die Durchführung eines Ad-hoc Usability-Tests und die Annotation, sondern der gesamte Prozeß des Konferenzaufbaus:

- Erprobung des Gesamtsystems im ISDN-Umfeld (2 B-Kanäle) mit unterschiedlichen Internet-Providern.
- Erprobung alternativer Audioübertragung mit Hilfe von rechnergestützter Telefonie.
- Erprobung des Mechanismus des Konferenzaufbaus.

**Keine Videoübertragung**

Aus den Erfahrungen der vorherigen Tests haben wir entschieden, die Audioübertragung über das Telefon zu realisieren. Auf eine Videoübertragung haben wir verzichtet, da zwei der drei Teilnehmenden nicht über die entsprechende Hardware verfügten und der Aufwand, die Hardware zu beschaffen, zu groß

war. Außerdem ist Video der Bestandteil des Systems, auf den wir am ehesten ohne wesentliche inhaltliche Einbußen verzichten können[93].

### 7.4.2 Durchführung des Tests

#### Vorbereitung

**Audioverbindung über Telefon**

Die Audio-Übertragung haben wir in diesem Test unter Einsatz herkömmlicher Technik umgesetzt. Über den ISDN-Konferenzmechanismus kann über einen ISDN-Kanal eine Dreierkonferenz mit Hilfe des Telefons geschaltet werden. Nur bei Etablierung der Dreierkonferenz sind zwei B-Kanäle nötig[94], nach dem Start wird die gesamte Konferenz über einen B-Kanal abgewickelt. Eine Alternative ist das neue Programm AVM-Fritz!Phone, das auf dem CAPI-Standard 2.0 aufsetzt.[95]

**Keine feste IP-Adresse**

Der Praxistest sollte über verschiedene Internet-Provider durchgeführt werden, diese waren:

- Universität Bremen (EntwicklerIn)
- Arcor Call-by-Call (ExpertIn)
- Vossnet (BenutzerIn)

**Internet-Call-by-Call**

Arcor erscheint uns als eine einfache Möglichkeit des Zugangs zum Internet, ohne eine Vertragsbindung mit einem Provider eingehen zu müssen. So kann jede und jeder mit entsprechender Hardware (Modem oder ISDN-Karte) über das Internet kommunizieren. Der Nachteil von Arcor wie auch von Vossnet ist, daß keine feste IP-Adresse zur Verfügung steht, sondern sie wird bei jedem Anmelden temporär zur Verfügung gestellt. Mit Hilfe von freien Tools (z. B. Quick-IP) ist es jedoch möglich, schnell die eigene IP-Adresse zu ermitteln. Dieser Umstand hatte zur Folge, daß neue Anforderungen an unser Systems entstanden, die den nächsten Versionen eingearbeitet werden müssen.

#### Durchführung

**Schritte des Konferenzaufbaus**

Der Konferenzaufbau wurde abweichend von unserer grundlegenden Konzeption wie folgt durchgeführt:

- Alle Teilnehmenden meldeten sich durch ihre Provider im Netz an. Der Benutzer (Marc Czaschke von der Firma Kohne) und der Entwickler (Manfred Wolff) übertrugen über E-Mail ihre zugewiesene IP-Adresse an den Experten[96] (Carsten Leßmann).

---

[93] Zur Diskussion über den Einsatz von Video vgl. ISAACS & TANG 1993, CHRISTEL et al. 1998, OLSON, OLSON & MEADER 1995, WEISKAMP 1996 und MOON 1998.

[94] Das ist außerdem davon abhängig, wie die Dreierkonferenz technisch umgesetzt ist. Herkömmliche Telefonanlagen erlauben es, auch während des Aufbaus nur einen B-Kanal belegen zu müssen.

[95] In der vorliegenden Version ist es nur möglich zu makeln; theoretisch können solche Tools aber auch eine Dreierkonferenz zur Verfügung stellen, da sie Teil des CAPI 2.0 Standards ist. Ein Kontakt mit AVM ergab, daß sie sich bemühen, die Dreierkonferenz in einer nächsten Version zu realisieren.

[96] Arcor bietet neben einem Call-by-Call Internet-Zugang auch einen freien E-Mail-Account an, der nach Beantragung innerhalb einer halben Stunde geschaltet wird.

- Der Experte bildete eine Konferenzkonfiguration[97] und übertrug sie mit Hilfe eines E-Mail-Attachements an die anderen Teilnehmenden.
- Die Konferenzkonfiguration wurde eingebunden, bevor ErgoNet gestartet wurde. Nach dem Starten mußte in der eigenen Konfiguration („Datei/Einstellungen“) die aktuelle IP-Adresse eingetragen werden.
- Der Experte lud die beiden anderen Teilnehmenden zur Konferenz ein, die ErgoNet-Konferenz konnte starten.

Nachdem der normale Telefonmechanismus kein Problem war, wurde eine Verbindung über Fritz!Phone erstellt. Auch diese Verbindung funktionierte einwandfrei, so daß sich die Beteiligten mit Hilfe von Headsets verständigen konnten. Der Vorteil einer Lösung mit einer Applikation wie Fritz!Phone ist, daß sich der Telefonbestandteil des Systems in die Konferenzsteuerung integrieren läßt.

### 7.4.3 Ergebnisse

Dieser Praxistest ergab folgende Ergebnisse, die Schlußfolgerungen für die Weiterentwicklung von ErgoNet in sich bergen:

- Über ein ISDN-Modem oder eine ISDN-Karte und Call-by-Call sind die technischen und organisatorischen Voraussetzungen für ErgoNet auf ein minimales Niveau gesunken.
- ErgoNet benötigt zusätzliche Mechanismen, um die Anforderung der temporär zugewiesenen IP-Adressen zu lösen. Eine Möglichkeit wäre die Anmeldung über einen *ILS*.[98]
- Mit Hilfe von Fritz!Phone oder vergleichbaren Softwareprodukten, die auf der CAPI-Schnittstelle aufsetzen, ist eine einwandfreie Tonübertragung gegeben.

Die letzte Erprobung hat noch einmal verdeutlicht, daß der Hardwareaufwand für einen Ad-hoc Usability-Test bei der EntwicklerIn und der BenutzerIn sehr minimal ist, und trotzdem sehr gute Ergebnisse zu erzielen sind.

[97] In weiser Voraussicht besteht die Konferenzkonfiguration im vorliegenden System aus einer Datei, die in das jeweilige ErgoNet-Verzeichnis kopiert werden kann.

[98] In MS-Netmeeting ist der ILS-Service bereits integriert (vgl. Kapitel 5.1).

# 8 Schlußbemerkungen

**Erfüllung der gesteckten Ziele**

Mit der vorliegenden Arbeit beschreiben wir die Entwicklung eines Systems, mit dem software-ergonomische Evaluationen unternommen werden können. Wir haben gezeigt, daß die eingesetzten Methoden geeignet sind, diesen Prozeß effektiv und effizient innerhalb des Software-Entwicklungszyklus durchzuführen.

**Nachweis in Praxistests und Bestätigung einer Notwendigkeit eines derartigen Systems**

Im Laufe der Präsentationen und Praxistests stellten wir fest, daß die Notwendigkeit eines derartigen Systems besteht und unser System diese zumindest teilweise befriedigen kann. Aufgrund der geringen technischen Anforderungen und der angepaßten Benutzungsschnittstelle ist es ohne großen Aufwand und umfangreiche Vorbereitungen möglich, eine Sitzung von ExpertIn, EntwicklerIn und BenutzerIn zur konkreten Problembesprechung zu initiieren.

**Erkennen weiterer Anwendungsmöglichkeiten**

Während der Entwicklung des Systems und durch die Ergebnisse aus den Praxistests haben wir realisiert, daß das System nicht nur zur Durchführung von Ad-hoc Usability-Tests geeignet ist, zu deren Unterstützung es eigentlich konzipiert wurde. Durch minimale Änderungen der Konfiguration (z. B. Verzicht auf die Integration von BenutzerInnen) können auch Ad-hoc-Inspektionen durchgeführt werden. Darüber hinaus ist es möglich, die Funktionalität des Systems zur direkten Unterstützung bei Benutzungs-problemen von Programmen (HelpDesk) einzusetzen. Auch hierbei müssen nicht alle drei Parteien eines Usability-Tests teilnehmen. Durch den Mechanismus der verteilten Applikationsdarstellung kann über das Medium Internet räumlich getrennt auf anderen Rechner kontrollierend eingegriffen werden, während parallel eine Dokumentation des Problems und seiner Behebung erstellt werden kann.

**Technische Probleme und Ausblick in die Zukunft**

Dabei ist zu beachten, daß die Mechanismen der Applikationsdarstellung, des Konferenzaufbaus und der Annotations-Erstellung ausgereift und einsatzfähig sind. Probleme werfen dagegen die Audio- und Videoübertragungen über das Internet auf, die beim heutigen Stand der Technik noch nicht praktikabel sind. Einerseits läßt die Qualität und der Funktionsumfang der verfügbaren Werkzeuge zu wünschen übrig, während andererseits die technischen Anforderungen zu hoch und damit nicht allgemein verfügbar sind. Wir sind aber der Meinung, daß sich dies in naher Zukunft ändern wird, so daß auf den Einsatz externer Geräte, wie beispielsweise das Telefon, verzichtet werden kann, da alle Datenübertragungen über das Internet als zusammenfassendes Medium geleitet werden können.

**ErgoNet als Anregungsfaktor für software-ergonomische Fragestellungen**

Über die technischen Aspekte hinaus haben wir festgestellt, daß der Themenkomplex Software-Ergonomie und insbesondere der Evaluationsbereich bei EntwicklerInnen positiv und interessiert aufgenommen wird. Die fachliche Qualifikation auf diesem Gebiet ist allerdings noch nicht fundiert. Daher bietet ErgoNet durch seine einfache Anwendung die Möglichkeit, sowohl EntwicklerInnen als auch BenutzerInnen für die Problematik der Software-Ergonomie zu sensibilisieren. Ein Lernprozeß kann angestoßen werden, der sich auf die Qualität der Softwareprodukte positiv auswirken kann.

**Umkehrung des normalen Entwicklungsprozesses**

Die Entwicklung des Systems im Hinblick auf einen präsentationsfähigen Prototypen auf der CeBIT'98 führte dazu, daß der herkömmliche Software-Entwicklungsprozeß umgekehrt werden mußte. Für den ersten Prototypen

wurden nach einer rudimentären Anforderungsermittlung alle technischen Möglichkeiten analysiert, die zum Einsatz kommen könnten. Hieraus gestalteten wir das System, daß auf der Messe präsentiert wurde. Erst nach einer Überarbeitung und der zweiten Präsentation erfolgte die fundierte Analyse des Gebiets Software-Ergonomie und Evaluation von Software-Systemen. Diese Herangehensweise bot uns den Vorteil, unsere praktische Erfahrung in der Erstellung von Software-Systemen in der ersten Phase dieser Arbeit einzusetzen, um die kurze Zeitspanne bis zum Messebeginn optimal zu nutzen. Anschließend erarbeiteten wir uns einen Zeitplan, der die Erstellung des theoretischen Fundaments erlaubte.

## Danksagung

Herrn Prof. Jürgen Friedrich danken wir für die Möglichkeit, diese Arbeit in Zusammenarbeit mit dem Institut für Softwareergonomie und Informationsmanagement beim TZI, anzufertigen. Im Verlauf der Arbeit erhielten wir hilfreiche Anmerkungen und praktische Unterstützung von Peter Ansorge, Guido Frick und Joachim Hinrichs. Aus dem Institut Digitale Medien und Netze wollen besonders Mathias Bock für seinen Einsatz im Rahmen der CeBIT98 danken. Die Praxistests konnten wir nicht ohne das Engagement der MitarbeiterInnen der Bremer Schule Bürgerweide und des Kohne Ingenieurbüros durchführen. Last but not least bedanken wir uns bei Uwe Haupt für die konstruktive Betreuung und die vielen hilfreichen Diskussionen und Anregungen.

## Literaturverzeichnis

[Andersen, Schmidt 1998] Andersen, F.-U., Schmidt, J.:
*Weltweit Video - MBone: Multimedia im Internet.*
In: c't magazin für computer technik 21'98, S. 262 - 268

[Ansorge, Haupt 1997] Ansorge, P., Haupt, U.:
*Ergonomie-Reviews und Usability-Testing als Beratungs- und Qualifizierungsinstrumente.*
In: Liskowsky, R., Velichkovsky, B.M., Wünschmann, W. (Hrsg.):
*Software-Ergonomie '97 Usability Engineering: Integration von Mensch-Computer-Interaktion und Software-Entwicklung.*
[Teubner] Stuttgart 1997 S. 55 - 69

[AOF 1998] Institut für Informatik, Universität-Freiburg:
*AOF-Authoring on the Fly.*
In: http://ad.informatik.uni-freiburg.de/mmgroup.aof.about 02.12.98 (liegt als Text vor)

[Beimel, Schindler, Wandke 1993] Beimel, J., Schindler, R., Wandke, H.:
*Wie Experten der Software-Ergonomie den Teil 10 (Dialogue Principles) der ISO 9241 bewerten.*
In: Rödiger, K.-H. (Hrsg.):
*Software-Ergonomie ¢93 Von der Benutzeroberfläche zur Arbeitsgestaltung.* [Teubner] Stuttgart 1993, S. 133 - 144

[Bevan 1994] Bevan, N.:
*Measuring usability as quality of use.*
In: ftp://ftp.npl.co.uk/pub/hci/papers/Measuring usability as quality of use.rtf 12.09.98 (liegt als Text vor)

[Bevan 1995] Bevan, N.:
*Human-Computer Interaction Standards.*
In: ftp://ftp.npl.co.uk/pub/hci/papers/HCI Standards.rtf 12.09.98 (liegt als Text vor)

[Bevan 1996] Bevan, N.:
*Integrating Usability into the Development Lifecycle.*
In: ftp://ftp.npl.co.uk/pub/hci/papers/Integrating Usability into Development Lifecycle.rtf 12.09.98 (liegt als Text vor)

[Bevan 1997] Bevan, N.:
*Quality and usability: A new framework.*
In: ftp://ftp.npl.co.uk/pub/hci/papers/qualusab.rtf 12.09.98 (liegt als Text vor)

[Bevan, Curson] Bevan, N., Curson, I.:
*Planning and Implementing User-Centred Design Using ISO 13407.*
In: ftp://ftp.npl.co.uk/pub/hci/papers/ucd-tut.rtf 12.09.98 (liegt als Text vor)

[Bevan, Kirakowski, Maissel 1991] Bevan, N., Kirakowski, J., Maissel, J.:
*What is Usability?*
In: ftp://ftp.npl.co.uk/pub/hci/papers/What is Usability.rtf 12.09.98 (liegt als Text vor)

[Bevan] Bevan, N.:
*Quality in use: incorporating human factors into the software engineering lifecycle.*
In: ftp://ftp.npl.co.uk/pub/hci/papers/Incorporating Human Factors into SW Lifecycle.rtf 12.09.98 (liegt als Text vor)

[Born 1990] Born, G.:
*Referenzhandbuch Dateiformate: Datenbanken, Tabellenkalkulationen, Text, Grafik, Multimedia, Sound und Internet.*
[Addison-Wesley] Bonn 1990, 5. Auflage 1997

[Branaghan 1997] Branaghan, R.J.:
*Ten Tips for Selecting Usability Test Participants.*
In: http://www.iwaynet.net/~besinc/testpart.htm 03.09.98 (liegt als Text vor), 1997

[Branaghan] Branaghan, R.J.:
*Testing, One -- Two -- Three: Fundamentals of Usability Testing.*
In: http://www.iwaynet.net/~besinc/fun_utesting.htm 03.08.98 (liegt als Text vor)

[Brooks 1994] Brooks, P.:
*Adding Value to Usability Testing.*
In: Mack, R.L, Nielsen, J.:
*Usability Inspection Methods.*
John Wiley & Sons Inc] New York, Chichester, Brisbane, Toronto, Singapore 1994, pp. 255 - 271

[Cakir, Dzida 1997] Cakir, A., Dzida, W.:
*International Ergonomic HCI Standards.*
In: Helander, M.G., Landauer, T.K., Prabhu, P.V. (Hrsg.):
*Handbook of the Human-Computer Interaction.*
[ELSEVIER] Amsterdam 1997, pp. 407 - 420

[Campbell 1992] Campbell, R.L.:
*Categorizing Scenarios: A Quixotic Quest?*
In: SIGCHI Bulletin 1992 Volume 24, No.4, pp. 16 - 17

[Christel et al. 1998] Christel, M., Smith, M.A., Taylor, R., Winkler, D.B.:
*Evolving Video Skims into Useful Multimedia Abstractions.*
In: Proceedings of the ACM CHI '98 (18. - 23. April 1998), pp. 171 - 178

[ComputerNews 1998]:
*Dataquest: Windows 98 wird Markt dominieren (18.06.1998).*
In: Suchergebniss aus http://www.computerwoche.de/info-point 15.12.1998 (liegt als Text vor)

[Computerwoche 1996]:
*Ovum analysiert HelpDesk-Boom.*
In: http://www.computerwoche.de/archiv/1996/02/C02GF07.SAU.html 03.12.98 (liegt als Text vor)

[Cuber, Wenzel 1994] Cuber, U., Wenzel, H.:
*Das Einmaleins der Windows-Programmierung.*
[Addison-Wesley] Düsseldorf, Wien 1994

[cyber-concepts]:
*Merz IT-Consulting - HelpDesk.*
In: http://www.cyber-concepts.de/html/helpdesk.html 03.12.98 (liegt als Text vor)

[Delphi 2 1996]:
*Borland Delphi 2, Das Buch.*
[Tewi Verlag] München 1996

[Desurvire 1994] Desurvire, H.W.:
*Faster, Cheaper!! Are Usability Inspection Methods as Effective as Empirical Testing.*
In: Mack, R.L, Nielsen, J.:
*Usability Inspection Methods.*
[John Wiley & Sons, Inc] New York, Chichester, Brisbane, Toronto, Singapore 1994, pp. 173 - 202

[Dix et al. 1995] Dix, A., Finlay J., Abowd, G., Beale, R.:
*Mensch Maschine Methodik*
[Prentice Hall] München 1995

[Duden 1997]
*Duden, das Fremdwörterbuch.*
[Bibliographisches Institut & F. A. Brockhaus] Mannheim 1997

[Ernst, Kottler 1996] Ernst, W., Kottler, J.J.:
*ActiveX Technik, Konzepte, Beispiele.*
[Markt&Technik] München 1996

[EU-Richtlinie 1990] EU-Richtlinie 90/270/EWG:
*Richtlinie des Rates vom Mai 1990 über die Mindestvorschriften bezüglich der Sicherheit und des Gesundheitsschutzes bei der Arbeit an Bildschirmgeräten (Fünfte Einzelrichtlinie im Sinne von Artikel 16 Absatz 1 der Richtlinie 89/391/EWG).* 1990

[Görner, Ilg 1993] Görner, C., Ilg, R.:
*Evaluation der Mensch-Rechner-Schnittstelle.*
In: Ziegler, J, Ilg, R. (Hrsg.):
*Benutzergerechte Software-Gestaltung: Standards, Methoden und Werkzeuge.*
[Oldenbourg] München, Wien 1993, S. 189 - 203

[Grady 1993] Grady, R.B.:
*Practical Results From Measuring Software Quality.*
In: Communications of the ACM/January 1993/Vol.36, No.11, pp. 46 - 47

[Gray et al. 1998] Gray, W.D., Atwood, M., Fisher, C., Nielsen, J., Caroll, J.M., Long, J.:
*Discount or Disservice? Discount Usability Analysis-Evaluation at a bargain price or simply damaged merchandise?*
In: Proceedings of the ACM CHI '98 (18. -23. April 1998), pp. 176 - 177

[Gunn 1995] Gunn, C.:
*An Example of Formal Usability Inspections in Practice at Hewlett-Packard Company.*
In: Interactive Posters of the ACM CHI '95 (07. - 11. Mai 1995), pp. 103 - 104

[Hacker 1994] Hacker, W.:
*Arbeits- und organisationspsychologische Grundlagen der Software-Ergonomie.*
In: Eberleh, E., Oberquelle, H., Oppermann, R. (Hrsg.):
*Einführung in die Software-Ergonomie.*
[de Gruyter] Berlin, New York 1994, S. 53 – 94

[Hammontree,Weiler, Nayak 1994] Hammontree, M., Weiler, P., Nayak, N.:
*Remote Usability Testing.*
In: Interactions Vol. 1, July 1994

[Hampe-Neteler, Rödiger 1992] Hampe-Neteler, W., Rödiger, K.-H.:
*Software-Ergonomie: Verfahren der Evaluierung und Standards zur Entwicklung von Benutzeroberflächen.*
In: Bericht Nr. 2/1992, Universität Bremen

[Hansen 1991] Hansen, M.:
*Ten Steps to Usability Testing.*
In: SIGDOC '91, Proceedings of the conference on 1991 ACM 9th annual international conference on systems documentations, pp. 135 - 139

[Hering 1992] Hering, E.:
*Software Engineering.*
[Vieweg] Braunschweig, Wiesbaden 1992, 3. Auflage

[Holdaway, Bevan] Holdaway, K., Bevan, N.:
*User System Intercation Standards.*
In: ftp://ftp.npl.co.uk/pub/hci/papers/Userneed.rtf 12.09.98 (liegt als Text vor)

[Holz auf der Heide 1993] Holz auf der Heide, B.:
*Welche software-ergonomischen Evaluationsverfahren können was leisten?*
In: Rödiger, K.-H. (Hrsg.):
*Software-Ergonomie ¢93 Von der Benutzeroberfläche zur Arbeitsgestaltung.*
[Teubner] Stuttgart 1993, S. 157 - 171

[Holzmann 1996] Holzmann, G.J.:
*On-The-Fly Model Checking.*
In: ACM Computing Surveys Vol. 28, No. 4es (Dezember 1996)

[Immler 1999] Immler, C.:
*Telefonieren übers Internet.*
In: PC Intern 01'99, S. 134 - 137

[Isaacs, Tang 1993] Isaacs, E., Tang, J.:
*What video can and can't do for collaboration: A case study.*
In: Proceeding of the conference on multimedia '93, pp. 199 - 206

[Jeffries et al. 1991] Jeffries, R.J., Miller, J.R., Wharton, C., Uyeda, K.M.:
*User interface evaluation in the real world: A comparison of four techniques.*
In: Proceedings of CHI '91 (New Orleans, LA, 28.4. - 3.5.1991), pp. 119 - 124

[John, Packer 1995] John, B.E., Packer, H.:
*Learning and Using the Congnitive Walkthrough Method: A Case Study Approach.*
In: Proceedings of the ACM CHI '95 (7. - 11. Mai 1995), pp. 429 - 436

[Kahn, Prail 1994] Kahn, M.J., Prail, A.:
*Formal Usability Inspections.*
In: Mack, R.L, Nielsen, J.:
*Usability Inspection Methods.*
[John Wiley & Sons, Inc] New York, Chichester, Brisbane, Toronto, Singapore 1994, pp. 141 - 171

[Kaminsky 1992] Kaminsky, S.K.:
*Test Early, Test Often.*
In: SIGDOC '92 Proceedings ofthe conference on 1991 ACM 10th annual international conference on systems documentations, pp. 47 - 55

[Karat C.-M. 1994] Karat, C-M.:
*A Comparison of User Interface Evaluation Methods.*
In: Mack, R.L, Nielsen, J.:
*Usability Inspection Methods.*
[John Wiley & Sons, Inc] New York, Chichester, Brisbane, Toronto, Singapore 1994, pp. 203 - 233

[Karat C.-M., Campbell, Fiegel 1992] Karat, C.-M., Campbell, R., Fiegel, T.:
*Comparison of Empirical Testing and Walkthrough Methodes in User Interface Evaluation.*
In: Proceedings of the ACM CHI '92 (03. - 07. Mai 1992), pp. 397 - 404

[Karat C.-M., Karat J. 1992] Karat, C.-M., Karat, J.:
*International Perspectives: Some Dialogue on Scenarios.*
In: SIGCHI Bulletin 1992 Volume 24, No.4, p. 7

[Karat J. 1997] Karat, J.:
*User-Centred Software Evalutation Methodologies.*
In: Helander, M.G., Landauer, T.K., Prabhu, P.V. (Hrsg.):
*Handbook of the Human-Computer Interaction.*
[ELSEVIER] Amsterdam 1997, pp. 689 - 704

[Karat J., Dayton 1995] Karat, J., Dayton, T.:
*Practical Education for Improving Software Usability.*
In: Proceedings of the ACM CHI '95 (7. - 11. Mai 1995), pp. 162 - 169

[Keil et al. 1998] Keil, M., Cule, P.E., Lyytinen, K., Schmidt, R.:
*A Framework for Identifying Software Project Risks.*
In: Communications of the ACM/November 1998/Vol.41, No.11, pp. 76 - 83

[Kensik, Prümper, Frese 1995] Kensik, A., Prümper, J., Frese, M.,:
*Ergonomische Gestaltung von Software auf Grundlage von handlungsorientierter Fehleranalyse.*
In: Böcker, H.-D. (Hrsg.):
*Software-Ergonomie ©95 Mensch-Computer-Interaktion Anwendungsbereiche lernen voneinander.* [Teubner] Stuttgart 1995, S. 217 - 232

[Knight, Myers 1993] Knight, J.C., Myers, E.A.:
*An Improved Inspection Technique.*
In: Communications of the ACM/January 1993/Vol.36, No.11, pp. 51 - 61

[Kyas 1996]: Kyas, O.:
*Internet professionell Technische Grundlagen & Praktische Nutzung.*
[International Thomson Publishing] Bonn 1996

[Kyng 1992] Kyng, M.:
*Scenario? Guilty!*
In: SIGCHI Bulletin 1992 Volume 24, No.4, pp. 8 - 9

[Lewis et al. 1990] Lewis, C., Polson, P., Wharton, C., Rieman, J.:
*Testing a walkthrough methodology for theory-based design of walk-up-and-use-interface.*
In: Proceedings of CHI '90 (Seattle, WA, 1.4. - 5.4.1990), pp. 235 - 242

[Lewis, Wharton 1997] Lewis, C., Wharton, C.:
*Cognitive Walkthroughs.*
In: Helander, M.G., Landauer, T.K., Prabhu, P.V. (Hrsg.):
*Handbook of the Human-Computer Interaction.*
[ELSEVIER] Amsterdam 1997, pp. 717 - 732

[Lindgaard 1994] Lindgaard, G.:
*Usability Testing and System Evaluation.*
[Chapman & Hall] London, Glasgow, New York, Tokyo, Melbourne, Madras 1994

[Maaß 1993] Maaß, S.:
*Software-Ergonomie Benutzer- und aufgabenorientierte Systemgestaltung.*
In: Informatik Spektrum Band 16 1993, S. 191 - 205

[Maaß 1995] Maaß, S.:
*Software-Ergonomie.*
In: Friedrich, J., Herrmann, Th., Peschek, M., Rolf, A. (Hrsg.):
*Informatik und Gesellschaft.*
[Spektrum] Heidelberg, Berlin, Oxford 1995, S. 222 - 238

[Mack, Nielsen 1994] Mack, R.L, Nielsen, J.:
*Executive Summary.*
In: Mack, R.L, Nielsen, J.:
*Usability Inspection Methods.*
[John Wiley & Sons, Inc] New York, Chichester, Brisbane, Toronto, Singapore 1994, pp. 1 - 23

[Mandl, Friedrich, Hron 1986] Mandl, H., Friedrich, H.F., Hron, A.:
*Psychologie des Wissenserwerbs.*
In: Weidenman, B., Krapp, A.:
*Pädagogische Psychologie.*
[Psychologie Verlags Union] München, Weinheim 1986, S. 145 - 218

[Masie 1996] Masie, E.:
*The Next Learning Trend: On-The-Fly.*
In: http://www.masie.com/onfly.html 02.12.98 (liegt als Text vor)

[Meyers Lexikon 1997]
*Meyers Lexikon in drei Bänden.*
[Bibliographisches Institut & F. A. Brockhaus] Mannheim 1997

[Microsoft 1995] Microsoft Cooperation:
*Die Windows-Oberfläche. Leitfaden zur Software-Gestaltung.*
[Microsoft Press] Unterschleißheim 1995

[Modzelewski 1998] Modzelewski, P.:
*Programming for Multiple Monitors in Windows 98.*
In: PC Magazine 07'98

[Moon 1998] Moon, Y.:
*The Effects on Distance in Local versus Remote Human-Computer Interaction.*
In: Proceedings of the ACM CHI '98 (18. - 23. April 1998), pp. 103 - 108

[Muller et al. 1995] Muller, M.J., McClard, A., Bell, B., Dooley, S., Meiskey, L., Meskill, J.A., Sparks, R., Tellam, D.:
*Validating an Extension to Praticipatory Heuristic Evaluation: Quality of Work and Quality of Work Life.*
In: Interactive Posters of the ACM CHI , 95 (07. - 11. Mai 1995), pp. 115 - 116

[Multimedia API 1997] Simon, R.J.:
*Multimedia, ODBC & Telefonie.*
[SAMS] München 1997

[Myers, Rosson 1992] Myers, B.A., Rosson, M.B.:
*Survey on User Interface Programming.*
In: Proceedings of the ACM CHI '92 (03. - 07. Mai 1992), pp. 195 - 202

[Nardi 1992] Nardi, B.A.:
*The Use of Scenarios in Design.*
In: SIGCHI Bulletin 1992 Volume 24, No.4, pp. 13 - 14

[NASA] NASA:
*Usability Testing Handbook DSTL-94-002.*
In: http://groucho.gsfc.nasa.gov/Code520/Code522/Documents/Usability/Handbook_....htm 03.08.98 (liegt als Text vor)

[NetmeetingSDK 1997] Microsoft Cooperation:
*Microsoft NetMeeting Software Developers Kit.*

[Neugebauer, Spielmann 1993] Neugebauer, C., Spielmann, N.:
*Das Ergonomie-Labor der SAP - empirische Ergonomie in der Praxis.*
In: Rödiger, K.-H. (Hrsg.):
*Software-Ergonomie ©93 Von der Benutzeroberfläche zur Arbeitsgestaltung.*
[Teubner] Stuttgart 1993, S. 327 - 330

[Nielsen 1992] Nielsen, J.:
*Finding usability problems through heuristic evaluation.*
In: Proceedings of the ACM CHI '92 (3. - 7. Mai 1992), pp. 373 - 380

[Nielsen 1993] Nielsen, J.:
*Usability Engineering.*
[Academic Press] Boston, San Diego, New York, London, Sydney, Tokyo, Toronto 1993

[Nielsen 1994a] Nielsen, J.:
*Heuristic Evaluation.*
In: Mack, R.L, Nielsen, J.:
*Usability Inspection Methods.*
[John Wiley & Sons, Inc] New York, Chichester, Brisbane, Toronto, Singapore 1994, pp. 25 - 63

[Nielsen 1994b] Nielsen, J.:
*Usability Laboratories: A 1994 Survey.*
In: http://www.useit.com/papers/uselabs.html 03.09.98 (liegt als Text vor)

[Nielsen 1995] Nielsen, J.:
*Usability Inspection Methods.*
In: Tutorials of the ACM CHI '95 (07. - 11. Mai 1995), pp. 377 - 378

[Nielsen F. 1996] Nielsen, F.:
*Human behavior: another dimension of standards setting.*
In: ACM-StandardView Vol. 4, No. 1 (March 1996), pp. 36 - 41

[Nielsen, Landauer 1993] Nielsen, J., Landauer, T.K.:
*A Mathematical Model of the Finding of Usability Problems.*
In: Proceedings of the ACM INTERCHI '93 (24. - 29. April 1993), pp. 206 - 213

[Nielsen, Molich 1990] Nielsen, J., Molich, R.:
*Heuristic evaluation of user interfaces.*
In: Proceedings of CHI '90 (Seattle, WA, 1.4. - 5.4.1990), pp. 249 - 256

[Nielsen] Nielsen, J.:
*Guerilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier.*
In: http://www.UseIt.com/papers/guerilla_hci.hm 03.08.1998 (liegt als Text vor)

[Olson J.S., Olson G.M., Meader 1995] Olson, J.S., Olson, G.M., Meader, D.K.:
*What Mix of Video and Audio is Useful for Small Groups Doing Remote Real-time Design Work?*
In: Proceedings of the ACM CHI '95 (7. - 11. Mai 1995), pp. 362 - 369

[Oppermann et al. 1992] Oppermann, R., Murchner, B., Reiterer, H., Koch, M.:
*Software-ergonomische Evaluation, Der Leitfaden EVADIS II.*
[de Gruyter] Berlin, New York 1992

[Oppermann, Reiterer 1994] Oppermann, R., Reiterer, H.:
*Software-ergonomische Evaluation.*
In: Eberleh, E., Oberquelle, H., Oppermann, R. (Hrsg.):
*Einführung in die Software-Ergonomie.*
[de Gruyter] Berlin, New York 1994, S. 335 - 371

[Paul 1995] Paul, H.:
*Exploratives Agieren.*
[Lang] Frankfurt am Main, Berlin, Bern, New York, Paris, Wien 1995

[PC Professional 1998]
*Tech Guide: Telefonieren im IP-Netz.*
[ZIFF-DAVIS-Verlag] Supplement PCpro 11/98

[Petzold 1996] Petzold, C.:
*Windows95 Programmierung.*
[Microsoft Press] Unterschleißheim 1996

[Pflüger 1992] Pflüger, J.P.:
*Usability Engineering Organisation der Kreativität im EDV-Projektmanagement.*
[Projekta Verlag] Winterthur 1992

[Pomberger, Blaschek 1996] Pomberger, G., Blaschek, G.:
*Software Engineering. Prototyping und objektorientierte Software-Entwicklung.*
[Hanser Verlag] München, Wien 1996, 2. Auflage

[Prosise 1997] Prosise, J.:
*IPC in Windows NT.*
In: PC Magazine 06'97

[Reisner 1992] Reisner, D.:
*Further Use of Scenario .*
In: SIGCHI Bulletin 1992 Volume 24, No.4, p. 15

[Reverse Engineering] TU-München
*Reverse Engineering.*
In: http://sunmedoc1.informatik.tu-muenchen.de:1111/Books/raasch/r-1.5.4.html 18.12.98 (liegt als Text vor)
Aus: Rasch, J.:
*Systementwicklung mit Strukturierten Methoden.*
[Hanser Verlag]

[Rieman et al. 1991] Rieman, J., Davies, S., Hair, C., Esemplare, M., Polson, P., Lewis, C.:
*An Automated Cognitive Walkthrough.*
In: Proceedings of CHI '91, (New Orleans, LA, 28.4. - 3.5.1991), pp. 427 - 428

[Rieman, Franzke, Redmiles 1995] Rieman, J., Franzke, M., Redmiles, D.:
*Usability Evaluation with the Cognitive Walktrough.*
In: Tutorials of the ACM CHI '95 (7. - 11. Mai 1995), pp. 387 - 397

[Rowley 1994] Rowley, D.E.:
*Usability Testing in the Field: Bringing the Laboratory to the User.*
In: Proceedings of the ACM CHI '94 (Boston 24. - 28. April 1994), pp. 252 - 257

[Rowley, David 1992] Rowley D.E., David G.R.:
*The Cognitive Jogthrough: A Fast-Paced User Interface Evaluation Procedure.*
In: Proceedings of the ACM CHI '92 (03. - 07. Mai 1992), pp. 389 - 395

[Rubenking 1997] Rubenking,. N.J.:
*COM Objects in Delphi, Part1.*
In: PC Magazine 07'97

[Sawyer, Flanders, Wixon 1996] Sawyer, P., Flanders, A., Wixon, D.:
*Making a Difference - The Impact of Inspections.*
In: Proceedings of the ACM CHI '96 (13. - 18. April 1996), pp. 376 - 382

[Scholles 1998] Scholles, F.:
*Soziologische und planungstheoretische Grundlagen - Plaungstheorie und methoden.*
In: http://www.laum.uni-hannover.de/ilr/lehre/Ptm/Ptm_Szenario.htm 24.11.98 (liegt als Text vor)

[Shneiderman 1987] Shneiderman, B.:
*Designing the User Interface, Strategies for effective Human-Computer-Interaction.*
[Addison Wesley] 1987

[Spool, Snyder, Robinson 1996] Spool, J.M., Snyder, C., Robinson, M.:
*Smarter Usability Testing: Practical Techniques for Developing Products.*
In: Proceedings of the CHI '96 conference companion on Human factors in computing systems: Common Ground, pp. 365-366

[Sullivan 1996] Sullivan, K.:
*The Windows®95 User Interface: A Case Study in Usability Engineering.*
In:
http://www.acm.org/sigchi/chi96/proceedings/desbrief/Sullivan/kds_txt.htm 03.08.1998 (liegt als Text vor), 1996

[Trauboth 1993] Trauboth, H.:
*Software Qualitätssicherung: konstruktive und analytische Maßnahmen.*
[Oldenbourg] München, Wien 1993

[Usability Management 1998]
*Understanding Usability.*
In: http://www.Usability.com/umi_what.html 03.08.98 (liegt als Text vor)

[Virzi 1997] Virzi, R.A.:
*Usability Inspection Methods.*
In: Helander, M.G., Landauer, T.K., Prabhu, P.V. (Hrsg.):
*Handbook of the Human-Computer Interaction.*
[ELSEVIER] Amsterdam 1997, pp. 705 - 715

[Wallmüller 1990] Wallmüller, E.:
*Software-Qualitätssicherung in der Praxis.*
[Hanser] München, Wien 1990

[Wandmacher 1993] Wandmacher, J.:
*Software-Ergonomie.*
[de Gruyter] Berlin, New York 1993

[Weiskamp 1996] Weiskamp, K. (Hrsg.)
*Desktop Video: Grundlagen, Technik, Standards.*
[Addison-Wesley] Bonn 1996

[Wharton et al. 1992] Wharton, C., Bradford, J., Jefrries, R., Franzke, M.:
*Applying Cognitive Walkthrougs to more Complexe User Interfaces: Experiences, Issues, and Recommendations.*
In: Proceedings of the ACM CHI '92 (03. - 07. Mai 1992), pp. 381 - 388

[Wharton et al. 1994] Wharton, C., Rieman, J., Lewis, C., Polson, P.:
*The Cognitive Walkthrough Method: A Practitioner's Guide.*
In: Mack, R.L, Nielsen, J.:
*Usability Inspection Methods.*
[John Wiley & Sons, Inc] New York, Chichester, Brisbane, Toronto, Singapore 1994, pp. 105 - 139

[Win32API 1996] Simon, R.J.:
*Windows95 API Bible - Win32 Programmierung.*
[SAMS] München 1996

[Wixon, Wilson 1997] Wixon, D., Wilson, C.:
*The Usability Engineering Framework for Product Design and Evaluation.*
In: Helander, M.G., Landauer, T.K., Prabhu, P.V. (Hrsg.):
*Handbook of the Human-Computer Interaction.*
[ELSEVIER] Amsterdam 1997, pp. 653 - 688

[Wottawa 1986] Wottawa, H.:
*Evaluation.*
In: Weidenman, B., Krapp, A.:
*Pädagogische Psychologie.* [Psychologie Verlags Union] München, Weinheim 1986, S. 705 - 733

[Wright 1992] Wright, P.:
*What s in a Scenario?*
In: SIGCHI Bulletin 1992 Volume 24, No.4, pp. 11 - 12

[Young, Barnard 1992] Young, R.M., Barnard, P.J.:
*Multiple Use of Scenarios: A Reply to Campbell.*
In: SIGCHI Bulletin 1992 Volume 24, No.4, p. 10

# Versicherung

Die vorliegende Diplomarbeit wurde von uns an der Universität Bremen, Fachbereich 3 Mathematik/Informatik, Studiengang Informatik, gemäß § 11 (7) der Diplomprüfungsordnung i. d. F. v. 22.12.1993 und § 9 der Studienordnung Informatik i. d. F. v. 01.10.1993 als Gruppenarbeit angefertigt.

Gemäß § 11 (3) der Diplomprüfungsodnung der Universität Bremen weisen wir, die Verantwortlichen, wie folgt aus:

**Praktischer Teil**

Carsten Leßmann

Annotation

Manfred Wolff

Konferenzsteuerung

**Schriftlicher Teil**

Carsten Leßmann

2.2, 2.5, 2.7, 3.3, 3.4, 5.1, 5.2, 5.3, 7.1

Manfred Wolff

2.1, 2.3, 2.4, 2.6, 2.8, 3.1, 3.2, 3.5, 5.4, 5.6, 7.2, 7.3, 7.4

Gemeinsame Kapitel

1, 5.5, 6, 8

Wir versichern, die Arbeit ohne fremde Hilfe erstellt und keine anderen als die angegebenen und gekennzeichneten Quellen und Hilfsmittel benutzt zu haben. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, sind als solche kenntlich gemacht.

Bremen, 19. Januar 1999

| Carsten Leßmann | ManfredWolff |
|---|---|