Universität Paderborn

Fachbereich Wirtschaftswissenschaften

Fach Wirtschaftsinformatik

Diplomarbeit

Analyse unterschiedlicher Clientkonzepte für kollaborative Umgebungen am Beispiel der prototypischen Implementierung einer Knowledge Management Plattform

vorgelegt von

Bernd Völlmeke

zur Erlangung des akademischen Grades

Diplom Wirtschaftsinformatiker

vorgelegt bei

Prof. Dr. Ludwig Nastansky

Prof. Dr. Leena Suhl

Paderborn, 30.09.2005

Abstract

In den vergangenen Jahren ist analog zur Bedeutung des Internets auch die Bedeutung des Browser als Thin-Client Benutzeroberfläche gewachsen. So sind browser-basierte Benutzeroberflächen heute nicht nur für die klassischen Web-Angebote, sondern auch für originäre Intranetanwendungen, die vorher mit spezialisierten Thick-Clients bedient wurden, verfügbar. Der Trend zu browser-basierten Benutzeroberflächen ist begründet durch die geringeren Kosten (TCO), die eine solche Anwendungsstruktur mit sich bringt. Diese überwiegen derzeit die potentiellen Nachteile, die vor allem auf einer schlechteren Usability und damit einhergehenden Produktivitätsverlusten basieren.

Die vorliegende Arbeit erklärt, wie die oben beschriebenen Vor- und Nachteile der verschiedenen Client-Konzepte zustande kommen und stellt darauf aufbauend ein neues Client-Konzept vor, mit dessen Hilfe versucht wird, die Vorteile beider bisheriger Client-Typen in einem Konzept zusammenzufassen, ohne die spezifischen Nachteile zu übernehmen. Darüber hinaus wird die Bandbreite der technischen Ansätze vorgestellt, welche zur Umsetzung dieses neuen Client-Konzepts derzeit propagiert werden.

Im praktischen Teil wird anhand der kollaborativen Anwendung einer elektronischen Wissenssammlung die prototypische Implementierung einer solchen Client-Komponente durchgeführt.

Stichwörter: IBM Workplace Managed Client, Rich Client, Thick Client, Thin-Client, SmartClient, Rich Client der 2. Generation, High Fidelity Client, Wissensdatenbank, K-Pool, TCO, Usability, Eclipse

Abstract

During the past ten years the importance of the browser as a thin-client user interface has grown analogues to the importance of the internet itself. Today, browser-based user interfaces are not only common for typical web-applications like online shopping for example, but spread throughout the whole spectrum of possible applications, particulary with regards to intranet-based bussiness application which traditionally were a domain of specialized thick-clients. The main reason for this shift of the focus towards browser-based thin-clients are the lower Total Costs of Ownership (TCO) generated by a thin-client application infrastructure. This cost advantages outweight the potential shortcomings of a degenerated usability of browser-based applications which leads to productivity disadvantages.

This thesis explains how the pros and cons of the different client concepts can be justified. Starting from that the thesis introduces a new client concept which tries to join the advantages of both client concepts without adopting the specific shortcomings of each concept. Further on, this thesis illustrates the spectrum of technical approaches propagated to be essential for the realization this new client concept.

The concrete prototype, which utilizes one of the preeceding approaches, will be a knowledge repository as an example for a distributed collaborative application.

Keywords: IBM Workplace Managed Client, Rich Client, Thick Client, Thin-Client, SmartClient, High Fidelity Client, Knowledge Repository, K-Pool, TCO, Usability, Eclipse

Inhaltsverzeichnis

1 Einleitung 1

1.1 Szenario und Motivation 1

1.2 Aufbau der Arbeit 2

2 Theoretische Grundlagen 5

2.1 Die Bandbreite verschiedener Client-Arten 5

2.1.1 Der Thick-Client 6

2.1.2 Der Thin-Client 7

2.2 Usability von Software 9

2.2.1 Teilkomponenten der Usability 11

2.2.1.1 Guessability 12

2.2.1.2 Learnability 12

2.2.1.3 Experienced User Performance 12

2.2.1.4 System Potential 13

2.2.1.5 Re-Usability 13

2.2.2 Design-Grundsätze 14

2.2.2.1 Consistency 14

2.2.2.2 Consideration of User Resources 15

2.2.2.3 Feedback 16

2.2.2.4 Error Prevention and Recovery 17

2.2.2.5 User Control 17

2.2.3 Möglichkeiten, den Nutzen von Usability zu bewerten und ihr Einsatz in Kosten-Nutzen Rechnungen 18

2.2.3.1 Quantifizierung der Kosten 20

2.2.3.2 Quantifizierung des Nutzens 21

2.2.3.3 Angewandte Investitionsrechnungsverfahren 25

2.3 Total Cost of Ownership 26

2.3.1 Aufbau eines TCO-Modells am Beispiel der Gartner Group 27

2.3.2 Übersicht über weitere TCO-Modelle 28

2.3.3 Ergebnisse verschiedener TCO Studien und Zusammenfassung 30

2.4 Wissensintensive Arbeitsumgebungen und Wissenssammlungen 32
2.4.1 Wissensintensive Arbeitsumgebungen 32
2.4.2 Wissenssammlungen 36

3 Bewertung von Clientkonzepten unter Berücksichtigung von Benutzergruppen eines spezifischen Anwendungsszenarios 39
3.1 Identifikation von Benutzergruppen für Wissensdatenbanken 39
3.1.1 Externe Benutzer 41
3.1.2 Wissensarbeiter 41
3.1.3 Mitarbeiter im Wissensmanagement 41
3.2 Kritik der vorgestellten Konzepte zur Bewertung der Vorteilhaftigkeit von IT-Investitionen 42
3.2.1 Kosten-Nutzen Rechnungen für Usability 42
3.2.2 Konventionelle TCO Modelle 44
3.3 Gegenüberstellung von Thick-Client und Thin-Client 47
3.3.1 Usability 47
3.3.2 Total Cost of Ownership (TCO) 50
3.4 Zusammenhänge zwischen Usability und TCO 52
3.4.1 Generelles Verhältnis zwischen Usability und TCO 52
3.4.2 Effizienzbetrachtungen 53
3.4.2.1 Bedeutung von Usability-Komponenten für verschiedene Benutzergruppen 54
3.4.2.2 Wichtigkeit von TCO-Aspekten für verschiedene Benutzergruppen 56
3.5 Rich-Clients der zweiten Generation als Lösungsansatz 57
3.5.1 Motivation und Einordnung von Rich-Clients der zweiten Generation 58
3.5.1.1 Browser-basierte Konzepte 59
3.5.1.2 Stand-Alone Rich-Clients der zweiten Generation 63
3.5.2 Eigenschaften von Stand-Alone Rich-Clients der zweiten Generation 65
3.5.3 Stand-Alone Rich-Clients der zweiten Generation als Benutzeroberfläche für Wissensdatenbanken 67

3.6 Diskussion: Welcher Client für welche Benutzergruppe 68

4 Das Eclipse RCP Framework und der IBM Workplace 73

4.1 Das Eclipse RCP Framework 73

4.2 IBM Workplace und der Workplace Managed Client 76

5 Vorstellung der implementierten Lösung 79

5.1 Der Knowledge-Pool - Eine Einführung 79

5.1.1 Einsatzszenario 80

5.1.2 Funktionalitäten 82

5.1.2.1 Grundfunktionen 82

5.1.2.2 Erweiterte Funktionalitäten 83

5.2 Vorstellung der WMC Komponente K-Pool 84

5.2.1 Aufbau 84

5.2.2 Funktionalität 86

5.2.3 Exkurs: Vom Eclipse Plugin zur Workplace Komponente 88

6 Fazit 95

7 Literaturverzeichnis 99

8 Eidesstattliche Erklärung 105

9 Anhang 107

Abkürzungsverzeichnis

ACL	Access Control List
API	Application Programming Interface
BSI	Bundesamt für Sicherheit in der Informationstechnik
DIN	Deutsches Institut für Normung e. V.
ERP	Enterprise Ressource Planning
EUP	Experienced User Performance
ISO	International Organization for Standardization
ISV	Independent Software Vendor
K-Object	Knowledge Object
K-Pool	Knowledge Pool
KM	Knowledge Management
NC	Network Computer
PC	Personal Computer
R&D	Research and Development
RC²	Rich Clients der zweiten Generation
RCO	Real Cost of Ownership
SSO	Single Sign On
TBO	Total Benefit of Ownership
TCO	Total Cost of Ownership
TEI	Total Economic Impact
TVO	Total Value of Ownership
TSTS	Time-Savings-Time-Salary
UI	User Interface
WDB	Wissensdatenbank
XML	Extended Markup Language

Abbildungsverzeichnis

Abbildung 2-1 - „Idealized learning curve illustrating the five components of usability“ 14

Abbildung 2-2 - Vorgehensmodell Usability-Maßnahme 20

Abbildung 2-3 - Typische Zahlungsreihe einer Usability Investition..... 24

Abbildung 2-4 - Übersicht über TCO Ergebnisse verschiedener Research Gruppen . 31

Abbildung 2-5 - Aufteilung von TCO auf ihre Basiskostenfaktoren 32

Abbildung 2-6 - Routine-Prozesse vs. Wissensintensive-Prozesse..... 35

Abbildung 2-7 - Ebenenbetrachtung von Wissenssystemen 37

Abbildung 2-8 - Wissensdimensionen die durch Wissenssammlungen unterstützt werden können 38

Abbildung 3-1 - Übersicht: Total Economic Impact (TEI) 47

Abbildung 3-2 - Simulation der Dialogbox-Funktionalität in Web-Anwendungen 49

Abbildung 3-3 - Bedeutung der Usability-Komponenten für verschiedene Benutzergruppen..... 56

Abbildung 3-4 - Kontinuum von Rich Clients der zweiten Generation..... 58

Abbildung 3-5 - Beispiel GMail - Registerkarten ohne Nachladen..... 60

Abbildung 3-6 - Zuordnung von Client-Arten zu Benutzergruppen und Einflußfaktoren 69

Abbildung 4-1 - Zweiteilung der Plugin-Struktur in Eclipse 74

Abbildung 4-2 - Aufbau Eclipse vor der Version 3.0 75

Abbildung 4-3 - Aufbau Eclipse seit der Version 3.0..... 76

Abbildung 4-4 - Übersicht Workplace-Konzeption 77

Abbildung 4-5 - Aufbau des Workplace Managed Client 78

Abbildung 5-1 - Aufbau des Knowledge Pools (K-Pool) 80

Abbildung 5-2 - Aufbau der WMC Komponente K-Pool 85

Abbildung 5-3 - Statusanzeige beim Download von Attachments..... 87

Abbildung 5-4 - Screenshot: Test als RCP Anwendung 89

Abbildung 5-5 - Screenshot: Test mit eigener Personality..... 90

Abbildung 5-6 - Screenshot: Test unter Nutzung des WMC Developer Toolkits 91

Tabellenverzeichnis

Tabelle 2-1 - Kategorien der Usability und ihre Messgrößen 10

Tabelle 2-2 - Mögliche Konstellationen bei der Bewertung von Usability-Maßnahmen 18

Tabelle 2-3 - Nutzenkategorien und Beispiele für zu schätzende Größen 23

Tabelle 2-4 - Gegenüberstellung TCO Modelle: Forrester - Gartner 29

Tabelle 3-1 - Ordnungskriterien zu Bildung von Benutzergruppen und deren Ausprägungen 40

Tabelle 4-1 - Rollenübersicht für Independent Software Vendors (ISV) 78

1 Einleitung

1.1 Szenario und Motivation

Im vergangenen Jahrzehnt hat sich das Internet sowohl im privaten aber vor allem auch im geschäftlichen Bereich von einer technologischen Randerscheinung für spezielle Berufsgruppen zu einem Massenphänomen entwickelt, das aus vielen Bereichen des täglichen Lebens nicht mehr wegzudenken ist. Parallel zu dieser Entwicklung fand auch eine Veränderung der Client-Anwendungen statt. Während seit dem Aufkommen der PCs in den 80er Jahren zunehmend auf die jeweilige Aufgabe spezialisierte Anwendungen (Thick-Clients) zum Einsatz kamen, wandelte sich dieser Trend, analog mit der zunehmenden Bedeutung des Internets, hin zu web-basierten Anwendungen (web-basierte Thin-Clients), die im Browser ausgeführt werden.

Die Gründe für diese Entwicklung bündeln sich in einem Begriff, welcher als der entscheidende Schwachpunkt von Thick-Clients angeführt wird: Total Cost of Ownership (TCO). Gemeint sind also, die Gesamtkosten die durch die Nutzung einer IT-Infrastruktur entstehen. In diesem Punkt werden web-basierte Thin-Clients, u.a. aufgrund der leichten Einführung und des geringeren Wartungsaufwands für die Clients, vielfach im Vorteil gesehen. Um die hierdurch erzielbaren Kosteneinsparungen zu realisieren wurden bis heute für viele ehemals Thick-Client basierte Geschäftsanwendungen alternative web-basierte Benutzerschnittstellen entwickelt oder die Thick-Clients komplett durch diese Benutzerschnittstellen ersetzt. Neuentwicklungen von Anwendungen wiesen in 2003 sogar zu 70% web-basierte Client-Konzepte auf.

Inzwischen zeigt sich jedoch, dass diese durch TCO-Überlegungen geprägte Fokussierung auf web-basierte Clients nicht für jede Art von Anwendung vorteilhaft ist, da bei der Entwicklung von klassischen Web-Anwendungen vorhandene Designbeschränkungen die Usability eines Softwareproduktes negativ beeinflussen können. Um diesen Usability Problemen von klassischen Web-Anwendungen zu begegnen und gleichzeitig deren TCO-Vorteile beizubehalten, wird derzeit an einer neuen Generation von Clients gearbeitet, welche die Usability-Vorteile von Thick-Clients mit den Betriebskosten von Web-Anwendungen verbinden soll.

Die vorliegende Ausarbeitung untersucht die spezifischen Vor- und Nachteile, welche die verschiedenen Client-Arten in Bezug auf Usability und TCO aufweisen. Darüber hinaus werden ein Überblick und eine Einordnung von Technologien gegeben, mit deren Hilfe die oben beschriebene nächste Generation von Clients realisiert werden können. Im praktischen Teil wird eine Knowledge-Management-Anwendung (als Beispiel für eine vernetzte Multiuser Anwendung) in Form einer Komponente auf einer bestehenden Plattform für Clients der nächsten Generation entwickelt.

1.2 Aufbau der Arbeit

Die Arbeit beginnt in Kapitel 2 mit der Erläuterung der später verwendeten Grundlagen. Hierzu gehören neben der Erklärung und Abgrenzung der Client-Arten (Abschnitt 2.1), die Definition der für diese Arbeit relevanten Aspekte der Usability (Abschnitt 2.2), sowie die Vorstellung der TCO-Problematik (Abschnitt 2.3). Kapitel 2 schließt mit einer Einführung in den Problembereich der wissensintensiven Arbeitsumgebung als Vorbereitung auf die praktische Aufgabenstellung (Abschnitt 2.4).

In Kapitel 3 werden zunächst Benutzergruppen für Wissenssammlungen identifiziert (Abschnitt 3.1). Anschließend werden die im Kapitel 2 vorgestellten Konzepte TCO und Kosten-Nutzen Rechnungen für Usability kritisch gewürdigt (Abschnitt 3.2). Nach einem Vergleich der beiden Client Konzepte Thin- und Thick-Client in Bezug auf TCO und Usability-Aspekte (Abschnitt 3.3) erfolgt eine Erläuterung des Verhältnisses von TCO und Usability sowie eine differenzierte Betrachtung der Wichtigkeit einzelner Aspekte dieser Themenbereiche für die identifizierten Benutzergruppen (Abschnitt 3.4). Abschnitt 3.5 stellt ein neues Client-Konzept, die Rich-Clients der zweiten Generation, als Lösungsansatz für die zuvor identifizierten Probleme vor. Hierbei werden sowohl unterschiedliche technische Realisierungsmöglichkeiten für Rich-Clients der zweiten Generation dargestellt und eingeordnet, als auch die Vorteile, die dieses Client-Konzept gerade für den Einsatz in Wissenssammlungen mit sich bringt, erläutert. Abschließend erfolgt die Zusammenfassung der im dritten Kapitel gewonnen Erkenntnisse in Form einer Diskussion, die darstellt, welche Client-Art jeweils für die identifizierten Benutzergruppen am zweckmäßigsten ist, und aufgrund welcher Faktoren eine Abweichung von dieser generellen Zuordnung sinnvoll sein kann (Abschnitt 3.6).

Kapitel 4 widmet sich der Vorstellung der für den Prototypen eingesetzten Technologie. In einem ersten Schritt wird das zugrunde liegende freie Framework[1] beschrieben, auf dem die letztlich verwendete Erweiterung aufsetzt (Abschnitt 4.1). Anschließend erfolgt die Darstellung dieser Erweiterung und eine kurze Einführung in die gesamte, mit dieser Erweiterung zusammenhängende Produktpalette (Abschnitt 4.2).

Die Vorstellung des Prototypen ist das Thema des 5. Kapitels. Da der Prototyp einem schon existierenden System nachempfunden ist, werden zunächst die Funktionen dieses bestehenden Systems erklärt und ihr Einsatz innerhalb der Organisation erläutert (Abschnitt 5.1). Danach befasst sich Abschnitt 5.2 mit der eigentlichen Implementierung des Prototypen. Hier werden sowohl der Aufbau und die Funktionalitäten der Anwendung dargestellt, als auch Besonderheiten bei der Entwicklung für diese neue Plattform dokumentiert.

Im Fazit (Kapitel 1) werden die gesammelten Erkenntnisse zusammengefasst wobei auch auf noch vorhandene Schwachstellen eingegangen wird, die als Einstiegspunkte für Folge-Projekte genutzt werden können.

[1] *für eine Definition sei auf Computerworld 2005 S.156 verwiesen*

2 Theoretische Grundlagen

In folgenden werden die Grundlagen der Arbeit vorgestellt. Zunächst werden die Begriffe Thin- und Thick-Client gegeneinander abgegrenzt und ihre Bedeutung im Rahmen dieser Arbeit präzisiert. Mit der Usability von Software und verschiedenen TCO-Modellen werden zwei unterschiedliche Konzepte für die Bewertung von Software bzw. IT-Infrastrukturen vorgestellt. Schließlich führt der Abschnitt über wissensintensive Arbeitsumgebungen im Allgemeinen und Wissenssammlungen im Speziellen, den Leser in den späteren praktischen Problembereich ein.

2.1 Die Bandbreite verschiedener Client-Arten

Die Diskussion um die geeignete Client-Art begann mit dem Aufkommen der PCs in den 80er Jahren. Während die IT-Infrastrukturen zuvor von Mainframe-Systemen[2] in Rechenzentren dominiert wurden und somit am Arbeitsplatz lediglich Terminals[3], also Thin-Clients, realisierbar waren, ermöglichten Personal Computer (PC) Client/Server Systeme mit reichen Anwendungsprogrammen[4]. Diese PC-basierten Thick-Clients setzten sich bis Mitte der 90er Jahre zunehmend durch. Gestoppt wurde deren Verbreitung erst mit dem Aufkommen des Internets. Nun rückte eine neue Art der Thin-Clients, die browser-basierten Anwendungen, in den Vordergrund und so werden heute über 70% aller neuen Anwendungsprojekte browser-basiert realisiert[5].

Für die Begriffe Thick- und Thin-Client existieren keine eindeutigen quantifizierbaren Ausprägungen, anhand derer eine scharfe Abgrenzung der beiden Client-Arten vorgenommen werden könnte. Darüber hinaus werden die Begriffe sowohl für die Clienthardware als auch für Clientsoftware eingesetzt. In den beiden nachfolgenden Abschnitten soll deshalb versucht werden, über qualitative Eigenschaften und Beispiele eine Abgrenzung der verschiedenen Client-Arten vorzunehmen, und den Einsatz der Begriffe im Rahmen dieser Arbeit zu präzisieren. Für eine Betrachtung der Vor- und

[2] *für eine Definition sei auf Voltz 1990 S.187 verwiesen*

[3] *für eine Definition sei auf Voltz 1990 S.307 verwiesen*

[4] *eine detaillierte Betrachtung der Geschichte von Benutzerschnittstellen findet sich bei Nielsen 1993 S.49ff.*

[5] *vgl. Miedl 2003*

Nachteile der Client-Konzepte in Bezug auf die in dieser Arbeit untersuchten Faktoren Usability und Total Cost of Ownership (TCO) sei auf Abschnitt 3.2 verwiesen.

2.1.1 Der Thick-Client

Alternativ zum Begriff Thick-Client werden die Begriffe Rich-Client, der vor allem die Reichhaltigkeit der Benutzerinteraktion betont, und Fat-Client, der die Hardwareausstattung des verwendeten Clients in den Vordergrund stellt, verwendet.

Als rein auf die Hardware beschränkter Begriff beschreibt ein Thick-Client ein an ein Netzwerk angeschlossenes Verarbeitungsgerät mit eigener (hoher) Rechenleistung, und der Möglichkeit, Daten in größerem Umfang lokal zu speichern. Ein solcher Client hat demnach einen eigenen Zustand der unabhängig von etwaigen Servern ist. Das klassische Beispiel für einen Hardware-Thick-Client ist der typische, mit dem Firmennetzwerk verbundene Arbeitsplatzrechner. Dieser verfügt über ein eigenes Betriebssystem, eine Rechenleistung, die durch klassische Büroaufgaben nur zu einem Bruchteil genutzt wird, und eine Festplatte, die zusätzlich zum Betriebssystem Daten (z.B. auch für lokal genutzte Programme) aufnehmen kann.

Die oben beschriebene Hardware bildet die Vorraussetzung für den Einsatz von Software-Thick-Clients. Im Bereich der Software bezieht sich Thick-Client auf die für eine bestimmte Client/Server Anwendung eingesetzte Client-Software. Typisch für Thick-Clients ist, dass die Verarbeitungslogik zu großen Teilen vom Client übernommen wird, während der Server meist nur als Datenquelle und Datenarchiv dient. Damit geht einher, dass im Vergleich zum Thin-Client Ressourcen der Client Hardware intensiver genutzt werden und eine lokale Installation der Client-Software notwendig wird. Eine weitere Eigenschaft von Software-Thick-Clients ist, analog zu der Betrachtung der Hardware Thick-Clients, eine gewisse Autarkie des Clients vom Server. So ist es mit Hilfe von Thick-Clients oft möglich, Arbeitsschritte unabhängig vom Server auszuführen und die Ergebnisse später mit dem Server abzugleichen.

In der Praxis existiert eine ganze Bandbreite verschiedener Thick-Clients mit jeweils unterschiedlichen Zielsetzungen. Eine Kategorie von Anwendungen, die sich eine Eigenschaft des Thick-Clients Konzepts, nämlich die Übernahme von Verarbeitungslogik durch den Client, explizit zu Nutze macht, sind Anwendungen für verteiltes Rechnen (z.B. *SETI@home*, *GIMPS*, *FightAids@home*). In diesem Bereich erübrigt sich prinzipbedingt auch die Diskussion um die geeignete Client-Art. Weitere Beispiele für Thick-Clients sind

E-Mail-Programme (z.B. *Microsoft Outlook*, *Mozilla Thunderbird*), Groupware[6]-Anwendungen (z.B. *Lotus Notes*) oder auch client-seitige Benutzerschnittstellen für ERP-Systeme (z.B. *SAP GUI*). Im Rahmen dieser Arbeit sollen Thick-Clients auf Clients eingegrenzt werden, die zur Darstellung von, in zentralen Datenquellen (insb. Wissenssammlungen) vorhandenen, Informationen dienen können.

2.1.2 Der Thin-Client

Ebenso wie beim Thick-Client existieren auch für das Konzept des Thin-Client unterschiedliche Begriffe, die teilweise stark durch Marketingeinflüsse geprägt sind. Während der Begriff *Lean Client* noch als unternehmensunabhängig angesehen werden kann, sind die Begriffe *Net-PC* oder *NC (Network Computer)* schon nahezu Produktnamen, die unterschiedliche technische Spezifikationen konkurrierender Firmen (*Microsoft*/*Intel* bzw. *Sun*/*Oracle*) beschreiben. Des Weiteren dienen Begriffe wie *Centralized-*, *Server-based-* oder *Controller-based Computing* als Umschreibung für das hinter Thin-Clients stehende Konzept.

Die Bezeichnung Thin-Clients für Hardware weist generell auf geringe Ressourcen hin. Allerdings ist die Schwankungsbreite bei Thin-Clients im Vergleich zum Thick-Client noch wesentlich größer. Die Ausprägungen von Thin-Clients reichen im Bereich der Arbeitsplätze von im Leistungsvermögen und um externe Schnittstellen reduzierte PCs (*Net PC*) bis hin zu Terminals, die an einem klassischen Mainframe Rechner betrieben werden. Allerdings ist der Begriff nicht auf den Arbeitsplatz beschränkt. So finden sich weitere Thin-Clients in Form mobiler Endgeräte wie PDAs oder funktionell erweiterten Mobiltelefonen (*Smartphones*).

Die Abgrenzung von Software-Thin-Clients in Client-/Server-Anwendungen ist gegensätzlich zu der des Software-Thick-Clients. Grundsätzlich übernehmen Software-Thin-Clients möglichst keine Verarbeitungslogik, sondern überlassen diese der Server-Anwendung. Eine genaue funktionale Abgrenzung der Aufgabenbereiche von Client und Server existiert allerdings nicht. Ein weiterer Versuch der Abgrenzung bezieht sich auf die Notwendigkeit der Installation auf den Clients. Ein Thin-Client ist demnach ohne explizite Installation nutzbar. Allerdings wird hierbei kritisiert, dass die Notwendigkeit einer

[6] *für eine Definition sein auf Broy 1999 S.352 verwiesen*

Installation, beispielsweise bei browser-basierten Clients, oft von der verwendeten Plattform abhängt. Zusätzlich findet eine Diskussion darüber statt, ob die Installation von Plugins als Installation im Sinne dieser Abgrenzung gesehen werden soll.

Thin-Client-Lösungen lassen sich grob in die folgenden drei Teilbereiche einordnen:

- Terminal-Dienste: Bei diesem Verfahren stellt der Server den Benutzern eine virtuelle Benutzeroberfläche zur Verfügung, die über spezielle Protokolle (z.B. *ICA* oder *X11*) auf dem Client dargestellt wird. Der Client überträgt seinerseits nur Benutzerinteraktionen (z.B. Mausbewegungen und Tastatureingaben) an den Server, der diese wiederum an die Anwendung übergibt. Das zu übertragende Datenaufkommen ist hierbei stark abhängig von der Art der genutzten Anwendung. *Nieh et al.* kommen in ihrer Untersuchung[7] zu dem Ergebnis, dass bei der Benutzung eines Browsers über ein solches System ein Datenaufkommen anfällt, welches zwischen 2 und 30 mal so hoch ist wie direkte Übertragung der Web-Seiten via HTML. Als Clients kommen hier sowohl Software-Clients für die verschiedene Systeme zum Einsatz, als auch dedizierte Hardware-Thin-Clients, deren einziger Zweck die Nutzung von Terminal-Diensten ist. Beispiele für Terminal-Systeme sind *Sun Ray*, *Citrix Metaframe* oder *Microsoft Terminal Services*.
- Mobiler Code: Unter diesem Oberbegriff sind Thin-Clients zusammengefasst, die als Anwendungen auf den Client übertragen werden und anschließend mit einem Server zusammenarbeiten. Dies kann entweder in Form von *Java-Applets* geschehen, die innerhalb eines Browser laufen, oder auch in entsprechend ausgelegten Umgebungen. Voraussetzung dafür ist, dass die Programmteile klein sind und alle Daten auf dem Server gespeichert werden, da die Hardware der Clients entsprechend des Thin-Client Paradigmas sehr eingeschränkt ist. Ein Beispiel einer solchen (in diesem Fall auf *Java* basierenden) Umgebung findet sich bei Kanter[8].
- Browser-Anwendung: Hier dient ein Internet-Browser als Schnittstelle zum Benutzer. Die Kommunikation zwischen Client und Server erfolgt daraus folgend über HTML. Beispiele für browser-basierte Anwendungen finden sich zahlreich im

[7] *vgl. Nieh et al. 2000 S.2*

[8] *vgl. Kanter 1998*

Internet; aber auch bei unternehmensinternen Businessanwendungen ist diese Art der Schnittstelle auf dem Vormarsch. So bieten nahezu alle namhaften Hersteller inzwischen browser-basierte Schnittstellen für ihre Produkte an.

Im Gegensatz zu (Software-)Thick-Clients, die aufgrund ihrer Anforderungen auf entsprechend ausgestattete Hardware angewiesen sind, fällt der Einsatz eine (Software-) Thin-Client Anwendung nicht zwangsläufig mit der Benutzung von Thin-Client-Hardware zusammen, sondern tritt oft auch in Kombination mit Thick-Client-Hardware auf (z.B. Software zur Nutzung von Terminaldiensten oder Nutzung von Browser-Anwendungen auf einem typischen PC). Im Rahmen dieser Arbeit sollen unter dem Begriff Thin-Client, aufgrund ihrer großen Bedeutung, browser-basierte Benutzerschnittstellen verstanden werden.

2.2 Usability von Software

Usability wird in der DIN ISO Norm 9241 Teil 11 mit dem Begriff „Gebrauchstauglichkeit“ übersetzt und als "das Ausmaß, in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und mit Zufriedenheit zu erreichen." [9] definiert. Diese allgemeine Definition bezieht sich zunächst auf Produkte jedweder Art; sie soll für die Zwecke dieser Arbeit aber auf die Usability von Softwareprodukten beschränkt bleiben.

Die Bedeutung der Usability hat im Bereich von Software-Produkten in den letzten Jahrzehnten durch die zunehmende Verbreitung von Computer-Systemen und der damit verbundenen Ausweitung des Benutzerkreises, sowie der Einführung grafischer Benutzeroberflächen rasant zugenommen[10]. Die Usability war zu Anfang der Entwicklung von Computern noch von untergeordneter Bedeutung, da aufgrund der damals eingeschränkten Leistungsfähigkeit der interaktive Betrieb nicht durchführbar war und somit der Programmierer oft auch den einzigen Benutzer darstellte. Diese Situation hat sich bis heute grundlegend geändert. Spätestens seit dem Durchbruch von kostengünstigen PCs in den 80er Jahren erfährt die Usability von Software eine stetig wachsende Aufmerksamkeit[11]. Bei heutiger Software stellt die Usability einen der wenigen Bereiche

[9] *vgl. DIN ISO 9241-11 1998*
[10] *vgl. Nielsen 1993 S.8 ff.*
[11] *vgl. Lindgaard 1994 S.3 f.*

dar, in denen sich ein Produkt von Konkurrenzprodukten positiv unterscheiden kann. Die zunehmende Beachtung von Usability-Aspekten zeigt sich sowohl bei privaten Konsumenten, die eine gute Bedienbarkeit inzwischen als selbstverständlich voraussetzen, als auch im geschäftlichen Umfeld, wobei Usability in diesem Bereich eher als Mittel zur Produktivitätssteigerung verstanden wird. Eine weitere Domäne, in der Usability eine entscheidende Rolle spielt, ist die Ausführung sicherheitskritischer Tätigkeiten. Als Beispiel werden in diesem Zusammenhang oft Atomkraftwerke, Züge oder Verkehrsflugzeuge genannt, bei denen eine Fehlbedienung ernste Konsequenzen hat. Diese Arbeit soll hauptsächlich auf die oben erwähnten Produktivitätsaspekte von Usability abstellen.

Wie kann der Grad der Usability eines (Software-)Produktes nun gemessen werden? Die oben zitierte DIN ISO Norm bezeichnet Usability als Summe von Effektivität, Effizienz und Zufriedenheit. Die strikte Trennung zwischen Effektivität und Effizienz ist bei praktischen Messungen nur schwer aufrechtzuerhalten, da sich beide Kategorien in ähnlichen Messgrößen widerspiegeln[12]. Trotzdem werden in der Literatur für jede einzelne dieser Kategorien Vorschläge für operationalisierte Messgrößen gemacht, die im praktischen Versuch mit (potentiellen) Benutzern der Software ermittelt werden können. Eine Übersicht dieser Messgrößen liefert Tabelle 2-1**.** Die im Rahmen dieser Arbeit hauptsächlich relevanten Kategorien Effektivität und Effizienz werden in den folgenden Ausführungen unter dem Oberbegriff Produktivität zusammengefasst.

Kategorie	**Messgröße**
Effektivität	Erfolgsquote Qualität der erbrachten Leistung
Effizienz	Anzahl der benötigten Schritte vs. minimal nötige Schritte Fehlerrate benötigte Zeit kognitive Arbeitsbelastung
Zufriedenheit	Qualitative oder quantitative Benutzerbefragung

Tabelle 2-1 - Kategorien der Usability und ihre Messgrößen

Nachfolgend wird zunächst der Begriff Usability in Teilkomponenten aufgespalten, um später eine detaillierte Zuordnung von einzelnen Usability-Komponenten zu Benutzergruppen vornehmen zu können. Daraufhin werden die für diese Arbeit relevanten

[12] *vgl. Lindgaard S.28 f.*

Prinzipien eines sinnvollen Benutzerschnittstellendesigns vorgestellt. Abschließend erfolgt eine Vorstellung von Verfahren, die es ermöglichen, den Wert von Usability(-Maßnahmen) monetär zu messen, um die Anwendung von betriebswirtschaftlichen Kosten-Nutzen-Rechnungen zu ermöglichen.

2.2.1 Teilkomponenten der Usability

Größen, welche die Usability (Benutzbarkeit), messen sind nicht statisch, sondern verbessern sich in unterschiedlicher Weise durch die zunehmende Erfahrung der Benutzer mit einem zunächst neuen (Software-)Produkt. Der durch Wiederholung der Tätigkeiten entstehende Lerneffekt kann demnach dazu führen, dass sich die eine zunächst festgestellte Produktivitätsreihenfolge für verschiedene Produkte im Laufe der Zeit verschiebt. Ein praktisches Beispiel für diesen Prozess aus dem IT Bereich ist das Ändern von Dateiendungen für eine Reihe von Dateien mit einer grafischen Oberfläche im Vergleich zu einer kommadozeilenbasierten Oberfläche. Während ein unerfahrener Benutzer mit Hilfe der grafischen Oberfläche die entsprechenden Befehle finden und ausführen kann - und diese dann für jede Datei wiederholt -, ist es im Falle der kommandozeilenbasierten Oberfläche fraglich, ob er die geforderte Operation ohne Hilfe überhaupt ausführen kann. Umgekehrt verhält es sich mit der Produktivität eines erfahrenen Benutzers, der das Ändern der Endungen in der Kommandozeile mit einem einzigen Befehl für alle Dateien ausführen kann, während er beim Umbenennen mit Hilfe der grafischen Oberfläche kaum schneller sein wird als der unerfahrene Benutzer.

Um diese Unterschiede in der Usability deutlich zu machen, spaltet *Jordan*[13] die Usability zunächst in drei (1991) und später in zwei weitere Komponenten auf (1994) die es ermöglichen, ein nach Erfahrung der Benutzer differenziertes Bild der Usability zu zeichnen[14]. Im Folgenden sollen diese fünf Komponenten kurz vorgestellt werden. Sie werden später in der Arbeit benutzt, um die differierenden Anforderungen verschiedener Benutzergruppen deutlich zu machen.

---

[13] *vgl. Jordan 1998 S.11 ff.*

[14] *eine ähnliche Aufgliederung von Usability in die fünf Attribute Learnability, Efficiency, Memorability, Errors und Satisfaction findet sich bei Nielsen 1993 S.26 ff.*

2.2.1.1 Guessability

Guessability bezieht sich auf den Aufwand, der bei der ersten Ausführung einer Aufgabe mit einem (unbekannten) Produkt entsteht. Messgrößen für die *Guessability* sind z.B. die Anzahl der Fehler, die Erfolgsquote oder die Zeit, die für diese Ausführung benötigt wird. Hieraus ergibt sich, dass diese Komponente der Usability vor allem in Bereichen wichtig ist, in denen der Anteil an Benutzern die ein Produkt nur einmalig nutzen, besonders hoch ist. Ein Mangel an *Guessability* wird diese Benutzer davon abhalten, das Produkt zu verwenden, da ihnen der nötige Aufwand für eine einmalige Nutzung zu hoch ist. Umgekehrt kann eine unzureichende *Guessability* aber auch bei häufiger Nutzung des Produktes durch einen Benutzer einen Wettbewerbsnachteil darstellen. Dies vor allem, weil Kaufentscheidungen häufig vom ersten Eindruck abhängen und eine auf den ersten Blick komplizierte Benutzung, selbst wenn sie langfristig sinnvoll ist, abschreckend wirken kann.

2.2.1.2 Learnability

Die *Learnability* misst den benötigten Aufwand, um die Bearbeitung einer Aufgabe zu erlernen. Hierbei wird der Aufwand der ersten Ausführung der Aufgabe nicht mitgezählt, da davon ausgegangen wird, dass die erste Ausführung mit besonderen Schwierigkeiten behaftet ist. Hieraus folgt, dass ein Produkt mit einer hohen *Learnability* dazu führt, dass Einarbeitungszeiten sehr kurz ausfallen. Dies ermöglicht es beispielsweise, dass viele Mitarbeiter in der Lage sind, bestimmte Aufgabe in einem Programm auszuführen und so Arbeitsabläufe flexibler gestaltet werden können. Auf diese Weise werden Abhängigkeiten von bestimmten (geschulten) Mitarbeitern vermieden, ohne weitere Schulungskosten zu verursachen.

2.2.1.3 Experienced User Performance

Mit *Experienced User Performance (EUP)* wird das Leistungsvermögen eines Benutzers bezeichnet, der zeitlich die *Learnability* Phase hinter sich gelassen hat. Gemeint ist also ein Benutzer, der langfristig mit einem Produkt arbeitet und gegebenenfalls entsprechende Schulungen für das Produkt erhalten hat. Meist wird davon ausgegangen, dass sich das Leistungsvermögen eines solchen Benutzers relativ stabilisiert hat bzw. sich allenfalls nur noch sehr langsam verändert. Die *EUP* ist demnach von besonderer Bedeutung, wenn das Produkt von den Mitarbeitern langfristig und häufig eingesetzt wird. Die relative Stabilität

der *EUP* führt dazu, dass ein durch eine mangelnde *EUP* begründeter Produktivitätsverlust, im Gegensatz zu den beiden zuvor genannten Komponenten *Guessability* und *Learnability,* nur schwer durch Gegenmaßnahmen (z.B. zusätzliche Schulungen) ausgeglichen werden kann.

2.2.1.4 System Potential

Das *System Potential* bezeichnet die theoretische, maximal erreichbare Produktivität eines Produktes unter der Vorraussetzung eines idealen Kenntnisstandes des Benutzers. Die *EUP* wird demnach durch das *System Potential* begrenzt, liegt in der Praxis aber meist deutlich unterhalb des *System Potentials*. Die Gründe hierfür sind vielfältig. So ist es beim Funktionsumfang heutiger Software-Pakete nahezu unmöglich, alle Befehle und die verschiedenen Möglichkeiten, diese aufzurufen, zu kennen. Andererseits wissen Benutzer um die Existenz von Funktionen (z.B. Tastaturkürzel), bevorzugen aber dennoch die für sie subjektiv angenehmeren Methoden (z.B. die Auswahl von Menüeinträgen). Wichtig wird das *System Potential* in Situationen in denen diese die *EUP* tatsächlich limitiert.

2.2.1.5 Re-Usability

Re-Usability bezieht sich auf den Effekt des Verlernens. Sie misst die Abnahme der Produktivität, wenn eine Aufgabe eine längere Zeit nicht mehr ausgeführt worden ist und nun erneut bearbeitet werden soll. Die konkrete Bedeutung von "längeren" richtet sich an dieser Stelle danach, ob der Zeitraum der Nichtausführung deutlich denjenigen Zeitraum überschreitet, in dem die Aufgabe sonst periodisch ausgeführt wurde. Die beiden theoretischen Extreme sind, dass sich die Produktivität nach einem gewissen Zeitraum wieder auf dem Stand der *Guessability*-Situation befindet oder im anderen Fall überhaupt nicht nachlässt, also auf dem letzten Stand der Nutzung verharrt. Ein Beispiel für eine Aufgabe, bei der eine hohe *Re-Usability* vorteilhaft ist, ist die Bedienung von Erfassungsgeräten für den Warenbestand und die Einspeisung der erfassten Daten in ein Warenwirtschaftssystem bei einer Inventur.

Nachfolgende Abbildung 2-1 fasst die Teilkomponenten der Usability zusammen und stellt sie sowohl gegeneinander, als auch zum zeitlichen Verlauf der Nutzung einer Software in Bezug.

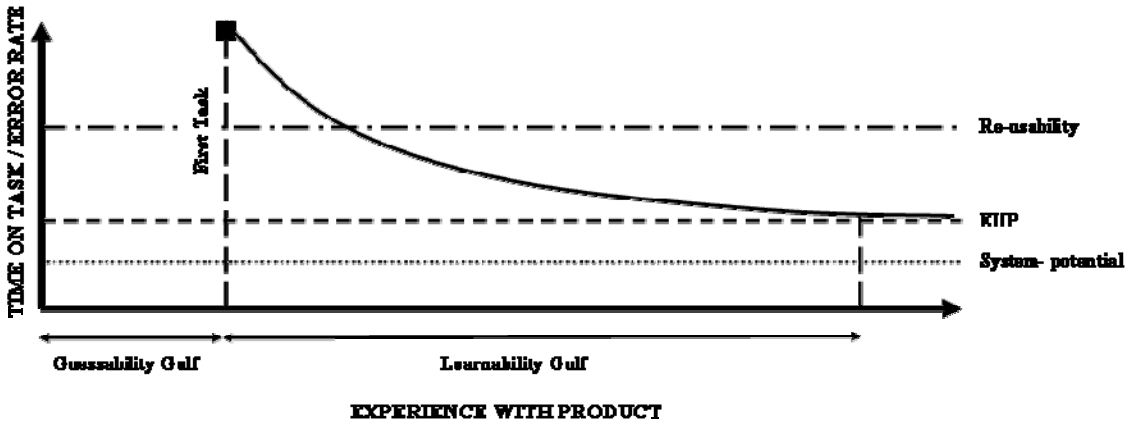


Abbildung 2-1 - „Idealized learning curve illustrating the five components of usability“[15]

2.2.2 Design-Grundsätze

Nachdem in den vorangegangenen Abschnitten die Wichtigkeit einer guten Usability sowie ihre Teilkomponenten dargestellt wurden, sollen in diesem Abschnitt einige Grundsätze vorgestellt werden mit denen eine gute Usability praktisch erreicht werden kann. In der Literatur finden sich zahlreiche, inhaltlich sehr ähnliche Ansätze, die versuchen, die wichtigsten Grundsätze möglichst allgemeingültig darzustellen[16]. Diese Ansätze unterscheiden sich zumeist nur in der Abgrenzung und der Anzahl der einzelnen Kategorien. Anschließend wird eine Teilmenge der Prinzipien eines sinnvollen Designs von *Jordan* vorgestellt, die geeignet ist, später Unterschiede zwischen den verschieden Client-Arten deutlich zu machen. Für eine vollständige und ausführlichere Beschreibung sei auf *Jordan*[17] verwiesen. Darüber hinaus werden zu jedem der vorgestellten Prinzipien Entsprechungen der DIN ISO Norm 9241 Teil 10[18], soweit vorhanden, aufgeführt.

2.2.2.1 Consistency

Eine konsistente Benutzeroberfläche ermöglicht die Generalisierung von gemachten Erfahrungen, um diese für die Bearbeitung neuer Aufgaben nutzbar zu machen. Hierbei können verschiedene Arten von *Consistency* unterschieden werden. Zunächst ist die

---

[15] *nach Jordan 1998 S.16*

[16] *vgl. beispielsweise: "Categories of typical usability defects" aus Lindgaard 1994 oder die “Usability Heuristics” aus Nielsen 1993 S.115ff.*

[17] *vgl. Jordan 1998 S.25 ff.*

[18] *vgl. DIN ISO 9241-10 1996*

Consistency innerhalb einer Anwendung zu nennen. Dies meint beispielsweise, dass spezielle Bearbeitungsschritte immer ähnlich ablaufen oder der konsistente Einsatz von Bezeichnungen und Symbolen beachtet wird. Des Weiteren gehört die Verwendung von über Programmgrenzen hinweg bestehenden Konventionen, sowohl innerhalb einer Programmart (z.B. Textverarbeitung) als auch darüber hinaus, in den Bereich von *Consistency*. Hierzu zählt z.B. die Verwendung und Anordnung der Menüs. Somit ist es möglich, in einer Anwendung erworbenes Wissen (z.B. das Öffnen einer Datei) nahezu ohne Lernaufwand auf eine andere Anwendung zu übertragen. Schließlich ist auch die Wahrung der Konsistenz zum benutzten Betriebssystem zu nennen. Hierzu zählt vor allem das so genannte "Look and Feel"[19] einer Benutzerschnittstelle; also die Verwendung von Design-Eigenschaften wie Schriftart und Schriftgröße bzw. Widgets[20] der jeweiligen Plattform. Eine weitgehende Entsprechung findet das Design-Prinzip der *Consistency* in der *Erwartungskonformität* und teilweise auch in der *Lernförderlichkeit* der DIN ISO Norm 9241 Teil 10.

2.2.2.2 Consideration of User Resources

Mit den Ressourcen oder Interaktionskanälen eines Benutzers werden bei diesem Design-Prinzip die Sinne bezeichnet, mit denen der Benutzer mit einem Produkt interagieren kann. Im Bereich des Computers sind hier vor allem die Augen (Monitor) und das Gehör (Lautsprecher) als Eingangskanäle und die Hände (Tastatur, Maus) als Ausgangskanäle zu nennen. Eine (noch) untergeordnete Rolle als Interaktionskanäle spielen der Tastsinn (Force Feedback) oder die Sprache (Spracherkennung bzw. Sprachsteuerung). Dieses Design-Prinzip besagt, dass die aktuelle Belastung der oben vorgestellten Interaktionskanäle beim Design von Anwendungen beachtet werden soll. Bei einer hohen Belastung eines Kanals soll für die Darstellung von weiteren Informationen auf einen weniger belasteten Kanal ausgewichen werden, oder dieser weitere Kanal zumindest genutzt werden, um die Aufmerksamkeit des Benutzers auf die neue Information zu lenken. Ein Beispiel, das in den meisten Computer-Systemen zu finden ist, ist die Benachrichtigung über den Eingang einer neuen E-Mail. Hier ertönt zusätzlich zu einem kleinen, in der Taskleiste erscheinenden Symbol, das durch die Belastung des visuellen

[19] *für eine Definition sei auf Greulich 2003 S.545 verwiesen.*

[20] *für eine Defintion sei auf Broy 1999 S.800 verwiesen.*

Kanals aufgrund anderer Tätigkeiten leicht übersehen werden kann, ein kurzer Hinweiston. So wird über einen zweiten Interaktionskanal die Ankunft einer neuen E-Mail signalisiert. Die zunehmend komplexer werdenden Benutzeroberflächen führen dazu, dass Forscher bemüht sind neue Kanäle der Interaktion für den Alltag nutzbar zu machen. Beispielhaft sei hier auf einen Beitrag von *Ian Oakley et. al.*[21] über die Nutzung von haptischer Interaktion in Benutzerschnittstellen verwiesen.

2.2.2.3 Feedback

Feedback bezeichnet die Reaktion des Systems auf eine Benutzeraktion. Prinzipiell unterscheidet man zwei Arten der Rückmeldung: Zum einen die direkte Rückmeldung auf Befehle die der Benutzer initiiert, so dass dieser erkennen kann, dass die Aktion vom System gestartet wurde bzw. ausgeführt wird, und zum anderen die Darstellung von Folgen, die ein Befehl des Benutzers hat. Ersteres verhindert beispielsweise, dass ein Benutzer Befehle mehrfach initiiert und dadurch Fehler hervorruft. Von besonderer Wichtigkeit ist hierbei die sofortige Bestätigung der Ausführung, bzw. bei länger andauernden Transaktionen die umgehende Information über den Status der Bearbeitung.

Bei der letztgenannten Art der Rückmeldung werden die implizite und die explizite Rückmeldung unterschieden. Die implizite Rückmeldung ist dadurch gekennzeichnet, dass ihre Bedeutung auch für unerfahrene Benutzer offensichtlich ist, z.B. die Darstellung von Formatierungen in Textverarbeitungssoftware. Die Interpretation der expliziten Rückmeldung erfordert hingegen ein gewisses Maß an Erfahrung, da die Bedeutung des jeweiligen *Feedbacks* zunächst erlernt werden muss. Als Beispiel führt *Jordan*[22] das Telefon an, bei dem die verschiedenen Zustände „Klingeln“ und „Besetzt“ durch verschiedene Tonfolgen repräsentiert werden. Wenn möglich, sollte gerade aufgrund der vielfältigen Möglichkeiten, die eine Benutzeroberfläche bietet, die implizite Art der Rückmeldung verwendet werden, um den Lernaufwand für den Benutzer möglichst gering zu halten. Die in den Grundsätzen der Dialoggestaltung geforderte *Selbstbeschreibungsfähigkeit* stimmt zu großen Teilen mit diesem Design-Prinzip überein.

[21] *vgl. Oakley et al. 2000*
[22] *vgl. Jordan 1998 S.29*

2.2.2.4 Error Prevention and Recovery

Dieser Design-Aspekt subsumiert mehrere Techniken der aktiven Fehlervermeidung und der Fehlerbehandlung. Zunächst spielt das Design bei der Vermeidung von Fehleingaben eine entscheidende Rolle. Darunter fällt sowohl die Verwendung von eindeutigen Bezeichnungen, die Vorgabe von sinnvollen Standardwerten als auch eine geeignete Anleitung des Benutzers. Letzteres geschieht zum Beispiel durch geführte Dialoge, die den Ablauf eines Arbeitsprozesses vorgeben und so die benötigten Eingaben in einen sinnvollen Zusammenhang stellen. Auf diese Weise wird dem Benutzer die Bearbeitung der Aufgabe erleichtert. Ein weiterer Faktor ist die umgehende Erkennung von Fehlern, um eine Speicherung von fehlerhaften Daten und daraus folgende Fehler zu vermeiden. Darüber hinaus muss der Benutzer auf Fehler in geeigneter Weise hingewiesen werden, so dass eine Korrektur - womöglich aufgrund eines geeigneten Korrekturvorschlags - leicht möglich ist. Ein Beispiel für diese Art von Fehlerkorrekturen ist die Validierung von Eingabefeldern bzw. das Verhindern des Abschlusses von Operationen mit ungültigen Daten. Ebenfalls zu den Funktionen, die Folgen von Fehlern minimieren, zählt die aus vielen Programmen bekannte Undo-Funktion, die es ermöglicht, irrtümlich ausgeführte Befehle rückgängig zu machen. Darüber hinaus kommt der Undo-Funktion eine besondere Bedeutung auch für andere Bereiche der Usability zu, da das Wissen um ihr Vorhandensein Benutzer zum „Erforschen“ einer Anwendung animiert. So können Benutzer ihnen bisher nicht bekannte Funktionen finden. In der DIN ISO Norm 9241 Teil 10 werden die hier beschriebenen Eigenschaften unter dem Begriff der *Fehlertoleranz* erläutert.

2.2.2.5 User Control

User Control bezeichnet als Anforderung einen hohen Freiheitsgrad der Interaktion des Benutzers mit der Anwendung. Dies bedeutet, dass dem Benutzer eine Auswahl von Möglichkeiten angeboten werden sollte, wie er seine Aufgaben durchführen kann. Hierzu zählt beispielsweise die Auswahl, ob ein Programm über die Maus oder über Tastaturkürzel bedient wird. Des Weiteren fällt die möglichst flexible Gestaltung von Dialogen und Assistenten in diesen Bereich. Hierdurch ergibt sich teilweise ein Zielkonflikt mit der oben beschriebenen Anleitungsfunktion der Assistenten, der je nach Erfahrungslevel der späteren Benutzergruppe gelöst werden muss. Ebenso verhält es sich mit der Anpassbarkeit der eigentlichen Benutzerschnittstelle, also der Anordnung der

verschiedenen Elemente der Benutzeroberfläche (z.B. Symbolleisten und Ansichten). Während eine hohe Anpassbarkeit für einen erfahrenen Benutzer eine große Erleichterung darstellen kann, in dem er seine Arbeitsoberfläche individuell auf seine Aufgaben abstimmt, wird eine große Anzahl von Einstellungsmöglichkeiten einen unerfahrenen Benutzer mitunter überfordern. Dies kann zu zusätzlichem Support-Aufwand führen, da eine irrtümlich angepasste Oberfläche wiederhergestellt werden muss. Sinnvolle, anpassbare Standardeinstellungen können helfen, dieses Problem zu mindern. Die oben dargelegten Anforderungen dieses Design-Prinzips finden in den Grundsätzen der Dialoggestaltung (DIN ISO 9241 Teil 10) in zwei getrennten Kategorien ihre Entsprechung. Die Kontrolle über den eigentlichen Arbeitsablauf wird dort mit der *Steuerbarkeit* bezeichnet, während Anpassungen der eigentlichen Oberfläche einer Anwendung unter der *Individualisierbarkeit* zusammengefasst werden.

2.2.3 Möglichkeiten, den Nutzen von Usability zu bewerten und ihr Einsatz in Kosten-Nutzen Rechnungen

Die Usability einer Software ist kein Selbstzweck, sondern dient, wie die Begriffe Effektivität und Effizienz in oben genannter Definition deutlich machen, dem Ziel, die Produktivität beim Einsatz von Software zu erhöhen. Die Produktivität wird hierbei durch die ebenfalls oben erwähnten Messgrößen ausgedrückt. Eine Erweiterung dieses Modells, das eine betriebswirtschaftliche Bewertung von Usability-Maßnahmen im Sinne von Investitionsentscheidungen ermöglicht, muss zusätzlich sowohl die Kosten der Usability als auch den festgestellten Nutzen quantifizieren. Hierdurch werden die Wirkungen der Usability-Maßnahmen in monetären Größen ausgedrückt und somit für klassische Investitionsrechnungsverfahren greifbar. In diesem Zusammenhang ist bei der Erhebung dieser Größen eine Differenzierung nach Art der Entwicklung und Art der Nutzung notwendig. In nachfolgender Tabelle 2-2 sind die insgesamt drei zu unterscheidenden Konstellationen dargestellt.

Entwicklung (Kosten)	Einsatz (Nutzen)	Bezeichnung der Rolle
Intern	Intern	Unternehmen entwickelt Software für Eigennutzung
Extern	Intern	Unternehmen kauft Software
Intern	Extern	Softwarehersteller

Tabelle 2-2 - Mögliche Konstellationen bei der Bewertung von Usability-Maßnahmen

Folgende Ausführungen konzentrieren sich, nach einigen grundsätzlichen Betrachtungen der Kostenseite, vor allem auf die Quantifizierung des Nutzens im Falle der internen Nutzung, da diese bei dem späteren Vergleich der Client-Arten im Vordergrund steht. Die Kostenseite wird in einem breiteren Ansatz im Abschnitt zu TCO behandelt.

Der Prozess der Kosten-Nutzen-Rechnung einer Usability-Maßnahme läuft iterativ ab. Zunächst werden die Kosten für eine festgelegte Usability-Maßnahme für eine Eigenentwicklung, bzw. die theoretischen Kosten der Usability eines zu beschaffenden Softwareproduktes bestimmt. *Mayhew und Mantei* schlagen vor, hierbei zunächst von einer maximalen Usability-Maßnahme auszugehen und anschließend eine sehr konservative Schätzung des durch die Maßnahme zu erwartenden Nutzens durchzuführen[23]. Wenn unter diesen Vorraussetzungen die Maßnahme als deutlich kostendeckend bewertet werden kann, ist der Prozess abgeschlossen. Ansonsten sollten zunächst die gemachten Nutzenschätzungen überprüft werden, um bei noch immer unbefriedigendem Ergebnis eine neue Iteration des Prozesses mit reduziertem Usability-Programm und entsprechend aktualisierten Nutzenschätzungen durchzuführen. Abbildung 2-2 illustriert dieses Vorgehen. Zu Kritik der hier vorgestellten Vorgehensweise sei auf Abschnitt 3.2.1 verwiesen.

[23] *vgl. Mayhew, Mantei 1994 S.38 f.*

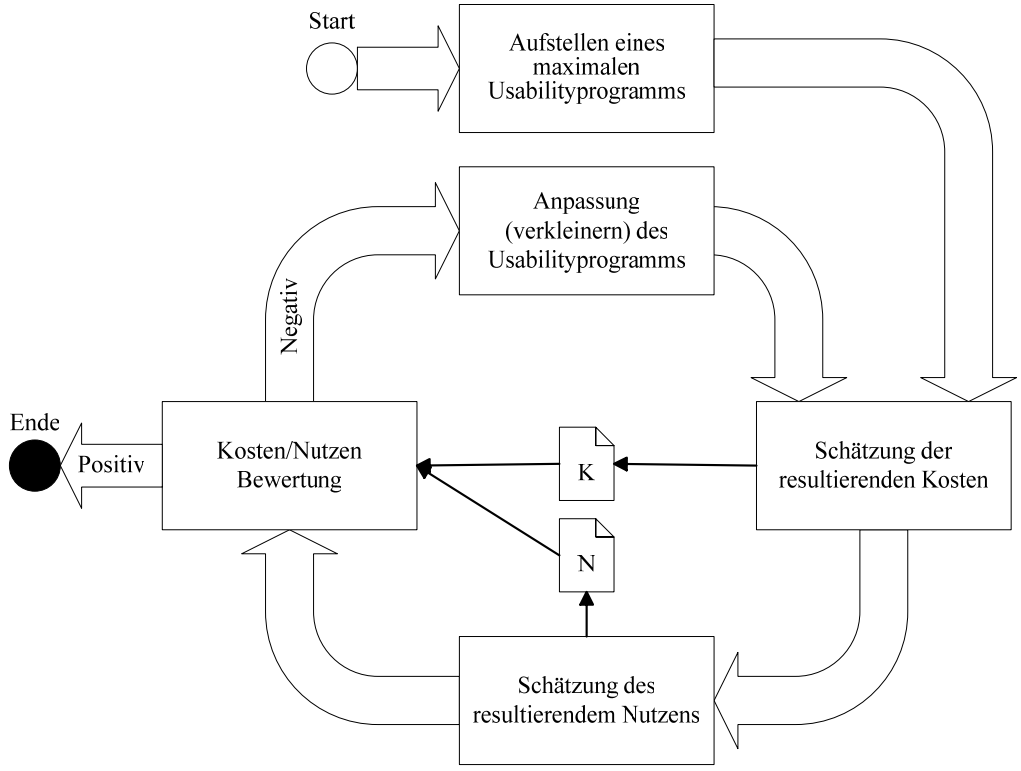


Abbildung 2-2 - Vorgehensmodell Usability-Maßnahme[24]

2.2.3.1 Quantifizierung der Kosten

Die Kosten der Usability definieren sich für die Seite, die Software entwickelt als diejenigen Kosten, die für das Usability-Programm entstehen, das die Entwicklung einer Software begleitet. Für eine Übersicht über Modelle eines Usability-Programms und die Beschreibung möglicher Bestandteile sei auf *Mayhew und Mantei*[25], *Lindgaard*[26] oder *Nielsen und Mack*[27] verwiesen. Zu diesem Zweck werden die einzelnen Maßnahmen eines Usability-Programms festgelegt und eine Schätzung über den Ressourceneinsatz einer jeden Maßnahme abgegeben. Diese Schätzung basiert für jede Maßnahme wiederum auf einer Unterteilung in einzelne benötigte Ressourcen. Im Rahmen des Usability-Programms durchgeführte Interviews mit gegenwärtigen und zukünftigen Benutzern werden beispielsweise in den Zeiteinsatz der Interviewer (Vorbereitung, Durchführung, Auswertung) und den der Interviewten aufgeteilt. Dieser wird jeweils mit den

[24] *Visualisierung des durch Mayhew, Mantei 1994 S.38f. vorgeschlagenen Vorgehens*
[25] *vgl. Mayhew, Mantei 1994 S.10 ff.*
[26] *vgl. Lindgaard 1994 S.12ff.*
[27] *vgl. Nielsen, Mack 1994*

Stundensätzen der beteiligten Personen bewertet und darüber hinaus die Kosten für andere Ressourcen (z.B. Raummiete, Video-Equipment zur Aufzeichnung) mit einbezogen. Die so ermittelten Kosten werden zunächst zu den Kosten der einzelnen Maßnahmen addiert. Diese werden dann durch Addition der Kosten jeder einzelnen Maßnahme zu Gesamtkosten des Usability-Programms aggregiert. Durch die Zerlegung in einzelnen Maßnahmen soll, analog zur Bildung von Arbeitspaketen im Projektmanagement, eine höhere Genauigkeit der Schätzungen erreicht werden. Darüber hinaus zwingt sie den Planenden, den Ablauf des Usability-Programms detailliert zu durchdenken und hilft so, die Kostenstruktur vollständig zu erfassen. *Mayhew und Mantei* empfehlen in der Praxis, die Schätzung der Kosten unter dem Gesichtpunkt der Vorsicht, also besser zu hoch als zu niedrig, durchzuführen, um die mit jeder Schätzung einhergehende Unsicherheit zu berücksichtigen[28]. Darüber hinaus sollte wie bei jedem Projekt ex post eine Analyse der zuvor kalkulierten und der tatsächlich angefallenen Kosten durchgeführt werden, um gesammelte Erfahrungen in zukünftige Schätzungen mit einfließen lassen zu können.

Bei der externen Software-Beschaffung, also dem Kauf von Software, lassen sich die Kosten, die durch Usability entstehen, erheblich schlechter abgrenzen. Zwar steht mit dem Kaufpreis bereits eine monetäre Größe zur Verfügung, in der sich verschiedene Software-Produkte unterscheiden, doch handelt es sich bei dieser Größe um einen Wert, der durch die verschiedensten Effekte (z.B. Funktionsumfang, Wettbewerbssituation) beeinflusst ist, so dass es schwer fällt, die Mehrkosten einer besseren Usability exakt zu identifizieren. Trotz dieser Schwierigkeit kann ein Vergleich des durch Usability entstehenden Nutzens helfen, Preisunterschiede zwischen verschiedenen Produkten zu erklären oder zumindest zu relativieren. Dies sollte neben anderen Kriterien (z.B. Funktionsumfang oder die Integration in die vorhandene IT-Infrastruktur) ebenfalls bei der Evaluation von Software in Betracht gezogen werden.

2.2.3.2 Quantifizierung des Nutzens

Die Quantifizierung des Nutzens von Usability-Maßnahmen identifiziert in einem ersten Schritt potentielle Bereiche in denen ein Mehrwert durch Usability entstehen kann. Hierbei ist die Unterscheidung zwischen interner und externer Nutzung zu beachten. Bei externer Nutzung, also dem Verkauf selbst entwickelter Software, werden zusätzlicher Umsatz und

[28] *vgl. Mayhew, Mantei 1994 S.9 ff.*

reduzierte Kosten bei, durch den Hersteller angebotenen, Support und Schulungen angeführt. Bei interner Nutzung der Software entsteht der Nutzen von Usability durch eine höhere Produktivität der Mitarbeiter und geringere Kosten bei ihrer Betreuung. Ein zusätzliches Einsparpotential bei interner Entwicklung ist es, unabhängig von der Nutzung der Software Einsparungen zu realisieren, die durch das frühzeitige Erkennen von Schwachstellen und die damit verbundene kostengünstigere Behebung dieser Schwachstellen einhergehen[29]. Eine weitere Unterteilung der oben identifizierten Kostenkategorien wird anschließend dahingehend vorgenommen, ob es sich bei diesen um einmalige Effekte, wie beispielsweise eingesparten Schulungsaufwand handelt, oder die einzusparenden Kosten periodisch über den Nutzungszeitraum der Software auftreten. Beispiele hierfür sind eine gesteigerte Produktivität oder eingesparte Kosten beim Support.

Nachdem die Bereiche für den zu untersuchenden Fall identifiziert und als einmalig bzw. periodisch klassifiziert wurden, erfolgt in einem nächsten Schritt die Quantifizierung der einzelnen Bereiche. Einen Sonderfall stellt die Quantifizierung der auf Usability-Maßnahmen zurückzuführenden Umsatzsteigerung dar, weil die Schätzung dieser Größe extrem schwierig ist. Die vielfältigen Einflüsse auf diese Größe führen dazu, dass selbst ex post kaum eine Zuordnung von Umsatzsteigerungen auf Produktmerkmale möglich ist, so dass hier zu Planungszwecken meist nur qualitativ argumentiert wird. Einen Anhaltspunkt für die Wichtigkeit von Usability-Aspekten kann eine Begutachtung von aktuellen Produkttests in diesem Umfeld geben. Allgemein empfehlen *Mayhew und Mantei*[30] eine gerade in diesem Punkt sehr vorsichtige Schätzung der erzielbaren Umsatzsteigerung. Die Quantifizierung der anderen Nutzenkategorien erfolgt meist auf der Basis von geschätzten Verbesserungen, die auf die Usability-Maßnahme zurückgeführt werden. Die Gewichtung dieser Verbesserungen wird anschließend mit der Häufigkeit der Durchführung und dem Stundensatz der durchführenden Mitarbeiter (TSTS-Verfahren) vorgenommen. Als Beispiel soll hier eine Rechnung zur Produktivität kurz vorgestellt werden[31]:

[29] *vgl. Stelzer 2002 S.45*

[30] *vgl. Mayhew, Mantei 1994 S.25 f.*

[31] *vgl. Mayhew, Mantei 1994 S.21*

Die folgende Schätzung geht davon aus, dass 250 Benutzer an 230 Arbeitstagen je 60 Eingabemasken bearbeiten. Die Eingabemaske kann durch Usability-Maßnahmen 1 Sekunde (1/3600 Stunde) schneller bearbeitet werden. Der Stundensatz der betroffenen Mitarbeiter beträgt 25$.
*250 * 60 * 230 * 1/3600 * 25$ = 23958$ Ersparnis/Jahr*

Ähnlich werden die Berechnungen zu anderen Nutzenkategorien durchgeführt. Die durch Usability-Maßnahmen eingesparte Zeit wird über die Stundensätze der Ausführenden in einen monetären Wert transformiert. Nachfolgende Tabelle 2-3 bietet einen Überblick über die auf diese Weise quantifizierbaren Nutzenkategorien und zeigt Beispiele für die zu schätzenden Größen auf.

Nutzenkategorie	**Anwend-barkeit**	**Auftreten**	**Beispiele für geschätzte Größen**
Produktivität	interne Nutzung	periodisch	Verringerung der Bearbeitungszeit Verringerung des Zeitaufwands für Fehlerbehebungen
Schulungsaufwand	interne / externe Nutzung	einmalig	intern: Verringerte Abwesenheit der Mitarbeiter vom Arbeitsplatz extern: Verringerung des Zeitaufwands für das Schulungspersonal
Supportaufwand	interne / externe Nutzung	periodisch	Verringerung des Zeitaufwands für das Supportteam und des Zeitaufwands der Mitarbeiter (nur intern)
Frühzeitiges erkennen von notwendigen Änderungen	nur bei interner Entwicklung	einmalig	Verringerter Zeitbedarf bei der Entwicklung

Tabelle 2-3 - Nutzenkategorien und Beispiele für zu schätzende Größen[32]

Wenn man davon ausgeht, dass die Werte für die Stundensätze der jeweiligen Mitarbeiter bekannt und Daten zur Durchführungshäufigkeit von bestimmten Tätigkeiten vorhanden sind, bzw. leicht ermittelt werden können, bleibt die Schätzung der jeweiligen Verbesserungen dieser Werte als größter Unsicherheitsfaktor. Diese werden auf Grundlage von Studien[33] und der Erfahrung des schätzenden Usability-Verantwortlichen getroffen. Hierbei ist zu beachten, dass wissenschaftliche Studien meist nur einen bestimmten Aspekt

[32] *in Anlehnung an Mayhew, Mantei S.20ff.*
[33] *für eine Übersicht über verschiedene Studien sei auf Mayhew, Mantei 1994 S.30f. verwiesen.*

der Benutzeroberfläche untersuchen und andere Aspekte in den entsprechenden Versuchen konstant gehalten werden um einen Effekt zu isolieren. Reale Umgebungen zeichnen sich hingegen dadurch aus, dass mehrere Faktoren im Rahmen eines Usability-Programms gleichzeitig verbessert werden. Hier wird empfohlen bei Vorliegen mehrerer, gleichzeitig verbesserter Aspekte einen Wert anzunehmen, welcher deutlich unterhalb der Addition aller Einzeleffekte, aber größer als der des größten Einzeleffekts ist. Diese Extrapolation der Ergebnisse einzelner Studien sollte, wie alle Schätzungen im Rahmen der Kosten-Nutzen-Analyse, von Vorsicht geprägt sein.

Schließlich erfolgt eine Aggregation, im Unterschied zu der Kostenseite jedoch zu einer Einzahlungsreihe, da die periodisch auftretenden Einsparungen in jedem Jahr der Software-Nutzung erneut auftreten. Typischerweise beginnt diese Zahlungsreihe mit einem hohen Wert und verharrt für den Rest der Nutzungsdauer auf dem entsprechend kleineren Wert der periodischen Einsparungen. Nachfolgende Abbildung 2-3 zeigt die idealisierte Zahlungsreihe einer solchen Usability Investition.

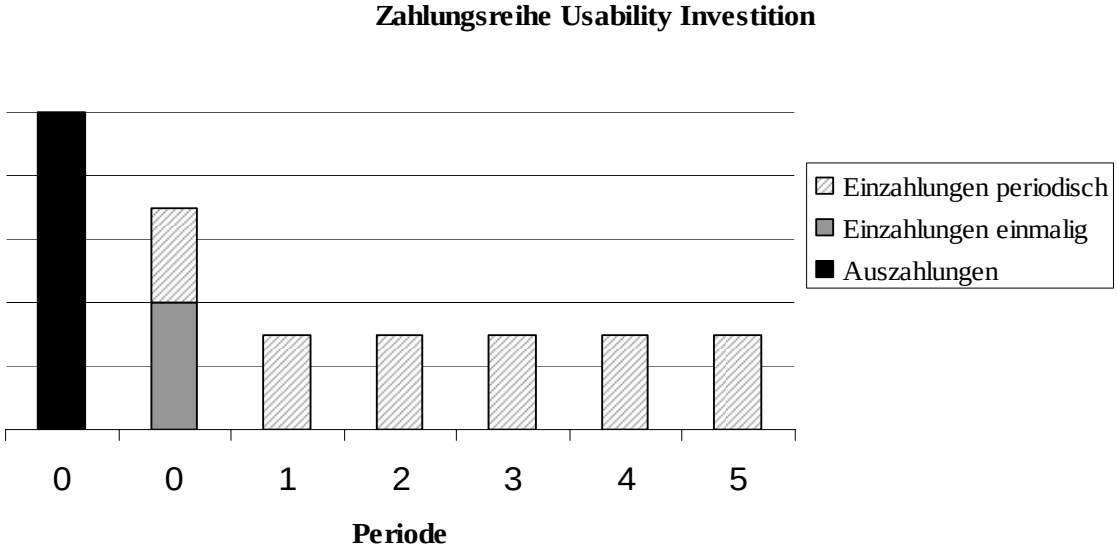


Abbildung 2-3 - Idealisierte Zahlungsreihe einer Usability Investition

Ebenfalls möglich ist eine degressiv verlaufende Zahlungsreihe, beispielsweise wenn die Einsparungen beim Support auf einen bestimmten Prozentsatz des Aufkommens geschätzt wurden und gleichzeitig von einem degressiven Verlauf des Support-Aufkommens ausgegangen wird, da sich dieses durch die zunehmende Erfahrung der Benutzer insgesamt verringert. Für eine erste Bewertung wird oft auch das kumulierte Einsparpotential über die

Nutzungsdauer angegeben, was allerdings, da die zeitliche Verschiebung nicht beachtet wird, zu ungenauen Ergebnissen führt.

2.2.3.3 Angewandte Investitionsrechnungsverfahren

Die in den vorangehenden Abschnitten durchgeführte Quantifizierung der Kosten und des Nutzens, die mit Usability zusammenhängen, ermöglicht es, Investitionen in Usability mit Hilfe betriebswirtschaftlicher Verfahren der Investitionsrechnung zu bewerten. Das letztlich verwendete Verfahren wird im konkreten Fall wohl eher durch die in der jeweiligen Organisation für andere Investitionen angewandte Praxis bestimmt. Grundlegend ist, dass mit den ermittelten monetären Werten Usability-Investitionen, trotz aller Unsicherheiten, rechenbar und somit konkret vergleichbar zu anderen Investitionen werden. Im Folgenden wird nun jeweils ein Beispiel für ein statisches und ein dynamisches Verfahren gegeben, welche die Verwendung der oben ermittelten Zahlungsreihen in der Investitionsrechnung verdeutlichen.

Armortisationszeit

Obwohl vielfach kritisiert[34], ist die statische Amortisationszeit aufgrund ihrer Einfachheit noch immer eine verbreitete und leicht zu vermittelnde Investitionsrechnung. Hierbei wird die initiale Auszahlung (Investition) den zu erwartenden Einzahlungen, im konkreten Fall also die erwarteten Einsparungen, gegenübergestellt und der Zeitpunkt bestimmt, an dem die kumulierten Einzahlungen die Auszahlung übersteigen. Der Zeitraum vor diesem Zeitpunkt wird dann als Amortisationszeit bezeichnet. Bessere Investitionen haben kürzere Amortisationszeiten. Die größten Kritikpunkte dieses Verfahrens sind zum einen die fehlende Betrachtung der Einzahlungen nach dem Amortisationszeitpunkt und zum anderen die Vernachlässigung des Zeitwertes des Geldes.

Kapitalwert und interne Zinsfußmethode

Die Kapitalwertmethode bezieht den Zeitwert des Geldes mit in die Berechnung ein, in dem sie die, durch Usability verursachten, Einzahlungen zukünftiger Perioden nur abgezinst berücksichtigt. Sie berücksichtigt also im Gegensatz zur Amortisationszeit sowohl alle Ein- und Auszahlungen, als auch die zeitliche Abfolge dieser. Grundlegend für diese Methode ist die Vorgabe eines Kalkulationszinssatzes der in der Berechnung

[34] *vgl. Heinhold 1989 S.72f. oder Däumler 1989 S.162ff.*

verwendet wird. Eine Investition wird demnach durchgeführt, wenn der mit dem Kalkulationszinssatz ermittelte Kapitalwert größer null ist. Der Kalkulationszinssatz gibt somit eine Mindestrendite an, die Investitionen erreichen müssen. Die auf dem gleichen Prinzip basierende, interne Zinsfußmethode geht den umgekehrten Weg. Bei ihr wird der Zinssatz bestimmt, der den Kapitalwert der Investition 0 werden lässt. Dieser Zinssatz bezeichnet dann die Rendite der Investition. Liegt er oberhalb des von Unternehmen vorgegebenen Vergleichszinssatzes, wird die Investition durchgeführt.

2.3 Total Cost of Ownership

Der Terminus Total Cost of Ownership (TCO) wurde im IT Bereich erstmals 1987 von der *Gartner Group* eingeführt. Die Analysten hatten erkannt, dass in bestehenden Kostenrechnungen der Werteverzehr, welcher aus der Nutzung der IT-Infrastruktur resultiert, meist nur aus dem Anschaffungsaufwand für Hard- und Software bestand. Die Kosten, die durch den Betrieb dieser Systeme entstanden, fanden keine Berücksichtigung. Das zu diesem Zeitpunkt noch auf Arbeitsplatzrechner beschränkte TCO-Modell wurde zunächst wenig beachtet. Das zugrunde liegende Konzept die gesamten Betriebskosten einer IT-Infrastruktur zu erfassen, erlangte jedoch in der Folgezeit, bedingt durch die in den 80er Jahren begonnene Neuausrichtung der Datenverarbeitung hin zu Client/Server-Systemen, zunehmende Bedeutung. Die Problematik war, dass trotz sinkender Hardwarekosten, die diese Dezentralisierung der IT-Infrastruktur erst ermöglicht hatte, die gesamten Betriebskosten der IT-Infrastruktur stiegen. Dies geschah vor allem aufgrund versteckter Organisationskosten, die ihren Ursprung in der mit der Umstellung einhergehenden Heterogenität und Komplexität der entstehenden neuen Infrastruktur hatten. 1996 wurden diese Kosten im Rahmen einer weiteren Studie der *Gartner Group*[35], erstmals umfassend für diese neue IT-Infrastruktur untersucht und Einsparpotentiale zwischen 25% und 40% aufgezeigt, die allein durch ein gezieltes Management einer IT-Infrastruktur realisiert werden können. In den folgenden Jahren wurden von unterschiedlichsten Analystengruppen (z.B. *Forrester, META Group*) TCO-Analysen durchgeführt und veröffentlicht. Die Ergebnisse dieser Analysen weisen aufgrund der unterschiedlichen Berechnungsgrundlagen und -ansätze eine hohe Schwankungsbreite auf. Da die verschiedenen TCO-Modelle zumeist auch Geschäftsgrundlage für den Verkauf von IT-Beratungsleistungen sind, werden die genauen Berechnungswege größtenteils nicht

[35] *vgl. Cappuccio et al. 1996*

veröffentlicht, so dass es schwer fällt, umfassende Informationen zu den verschiedenen Modellen zu bekommen. Dementsprechend wird im folgenden Abschnitt zunächst das TCO-Modell der *Gartner Group* vorgestellt, da es das bekannteste und am besten dokumentierte Modell ist. Anschließend werden Unterschiede zu anderen Modellen aufgezeigt und eine Übersicht über die Ergebnisse der verschiedenen Berechnungen gegeben.

2.3.1 Aufbau eines TCO-Modells am Beispiel der Gartner Group

Das TCO-Modell der *Gartner Group* beinhaltet eine detaillierte Aufschlüsselung der im Zusammenhang mit der IT-Infrastruktur anfallenden Kosten. Diese ermöglicht in einem ersten Schritt die vollständige Erfassung der anfallenden Kosten und in einem Zweiten den Vergleich der so ermittelten Kosten mit Referenzmodellen bzw. ähnlichen Organisationen. Durch den Vergleich dieser Werte können Potentiale in der eigenen IT-Infrastruktur aufgedeckt und durch entsprechend angelegte Maßnahmen genutzt werden. Der grundlegend neue Ansatz des TCO-Modells ist die Berücksichtigung von indirekten Kosten, die durch den Einsatz der IT entstehen. Im TCO-Modell zeigt sich dies in einer Zweiteilung der zu erfassenden Kosten in *budgetierte Kosten* und die so genannten *unbudgetierte Kosten*. Letztere stellen die indirekten Kosten dar und ergeben sich aus "effizienzhemmenden Vorgängen im Rahmen der Nutzung einer IT-Infrastruktur"[36]. Unterhalb der Einteilung in *budgetierte* und *unbudgetierte Kosten* werden wiederum verschiedene Kostenbereiche unterschieden wie z.B. der Bereich der Anschaffungskosten für Hard- und Software innerhalb der *budgetierten Kosten,* oder der Produktivitätsausfall als Bereich der *unbudgetierten Kosten*. Die beiden Bereiche sind durch weitere konkrete Kostenarten unterteilt, die wiederum nach unterschiedlichen Aspekten mehrfach kategorisiert werden. Für eine Übersicht über ein konkretes, nach dem Ansatz der *Gartner Group* entworfenes TCO-Modell sei auf den Anhang (a) verwiesen.

Eine besondere Schwierigkeit bei der Erfassung der Kosten stellt neben der Vielzahl der Kostenarten, die letztlich aber auch einen der Vorteile des Modells begründet, die Erfassung der *unbudgetierten Kosten* dar. Obwohl das Modell die Berücksichtigung dieser als wichtigste Neuerung vorsieht, werden keine spezifischen Ansätze zur Erhebung

[36] *vgl. Wild, Herges 2000 S.11*

vorgestellt, sondern lediglich allgemein auf bekannte Möglichkeiten der Erhebung durch Fragebögen oder (Gruppen-)Interviews verwiesen.

Die Ergebnisse einer solchen TCO-Analyse schwanken, bedingt durch sowohl zeitliche Einflüsse in denen die verschiedenen Analysen durchgeführt wurden, als auch durch die kontinuierliche Weiterentwicklung und den damit einhergehenden Schwerpunktverlagerungen des Modells. Für einen Desktop-Computer gibt die *Gartner Group* den jährlichen TCO-Wert mit ca. 7000$ bis 13000$ an. Hierbei ist zu beachten, dass TCO-Werte grundsätzlich nur wie beim oben erwähnten Vergleich differenziert verglichen werden sollten, und die hier angegeben Werte allenfalls eine Tendenz beschreiben können.

2.3.2 Übersicht über weitere TCO-Modelle

Neben der *Gartner Group*, haben sich auch andere Analystengruppen der TCO-Thematik angenommen und eigene Modelle zur Berechnung der TCO entwickelt. Sie weisen im Prinzip eine ähnliche Struktur auf, beziehen aber je nach Schwerpunktbildung einzelne Kostenfaktoren unterschiedlich stark ein, oder ziehen Abgrenzungen von IT-Infrastrukturkosten entsprechend breiter oder enger. Dieser Heterogenität entsprechend und auch aus Marketinggründen existieren unterschiedliche Bezeichnungen für die Modelle verschiedener Gruppen (z.B. *RCO - Real Cost of Ownership* - Modell der *META Group*).Beispielhaft werden im Folgenden die Unterschiede zwischen dem Modell der *Gartner Group* und zwei konkurrierenden Modellen von *Forrester Research* und der *META Group* vorgestellt.

Forrester Research

Dieses Modell teilt die anfallenden Kosten in so genannte Kostenfaktoren auf. Diese Kostenfaktoren lassen sich weitgehend auf die im *Gartner* TCO-Modell verwendeten Kostenkategorien abbilden (vgl. Tabelle 2-4).

Forrester Research (Kostenfaktoren)	**Gartner Goup** (Kostenkategorien als Kostenfaktoren interpretiert)
IT-Infrastruktur konstituierende Hard- und Software	Kostenkategorie Hard- und Software
Wartungsverträge	Kostenkategorien Operations und Verwaltung
Management einer IT-Infrastruktur	Kostenkategorien Operations und Verwaltung
Support Dienste	Kostenkategorie Operations
Mittelbar aus Nutzung einer IT-Infrastruktur hervorgehende Aktivitäten	Kostenkategorien Verwaltung und End-User-Operations
Zeiten, in denen Teile einer IT-Infrastruktur von ihren Anwendern nicht nutzbar sind	Kostenkategorie Downtime
Die eine Disaster-Vorsorge und ein Disaster-bedingtes Recovery umfassenden Aktivitäten	Kostenkategorien Operations und End-User-Operations

Tabelle 2-4 - Gegenüberstellung TCO Modelle: Forrester - Gartner[37]

Diese Kostenzuordnungen weisen in den meisten Kategorien nur Abweichungen in detaillierten Unterpunkten auf, stimmen also inhaltlich weitgehend überein. Eine Ausnahme bildet der Bereich der effizienzhemmenden Vorgänge die aus der Nutzung der IT-Infrastruktur entstehen. In diesem Punkt weist das Modell der *Gartner Group* eine deutlich höhere Anzahl von (Teil-)Kostenfaktoren auf. Der Grund hierfür ist, dass *Forrester Research* den im *Gartner*-Modell berücksichtigten *Futz-Faktor* nicht in ihr Modell aufnimmt. Der *Futz-Faktor* war von der *Gartner Group* als Teilkostenfaktor eingeführt worden. Unter ihm sind die Kosten zu subsumieren, welche durch unproduktiven Einsatz der IT-Infrastruktur eines Unternehmens entstehen (z.B. private Benutzung des Internets während der Arbeitszeit, Spiele etc.). *Forrester Research* führt hauptsächlich zwei Argumente an, die gegen einen Einbezug dieses Kostenfaktors in den TCO-Wert sprechen. Zum einen existiert keine Möglichkeit, die durch *Futzing* entstehenden Kosten zuverlässig zu beziffern, und zum anderen scheint dieser Faktor, auch aufgrund seiner schwierigen Erhebung und der dadurch erschwerten Erfolgskontrolle durch das verantwortliche IT-Personal, kaum beeinflussbar. Eine Einbeziehung des *Futz-Faktors* erhöht demnach den TCO-Wert nur um einen bestimmten, sehr wage geschätzten Betrag. Dieser verbessert weder dessen Aussagekraft, noch liefert er Informationen, die Grund für ein steuerndes Eingreifen sein könnten.

[37] *vgl. Wild, Herges 2000 S.17*

META Group

Einen noch restriktiveren Ansatz verfolgt das TCO-Modell der *Meta Group*. Das als *RCO (Real Cost of Ownership)* bezeichnete Modell bezieht grundsätzlich nur Kosten in die Berechnung ein, die aus der Nutzung der IT-Infrastruktur entstehen und die durch Belege erfasst werden können. Diese entsprechen den *budgetierten Kosten* des TCO-Modells der *Gartner Group*. Allerdings ist die Gliederung dieser Kosten grundlegend anders. Insbesondere die einzelnen Kategorien für *Migrationskosten* und *Legacy Systeme* zielen bereits auf die Identifikation von konkreten Potentialen einer IT-Infrastruktur ab. Insgesamt unterscheidet das *RCO*-Modell folgende fünf Kostenkategorien für die direkten Kosten:

- Allgemeine Anwenderkosten (Hard- und Software, Schulung, Wartung, Zinszahlungen)
- Kosten aus dem Betrieb von ERP-Systemen
- Kosten aus der Aufrechterhaltung von Netzwerkstrukturen (z.B. LAN, WAN, Telefon)
- Migrationskosten
- Kosten die aus der Aufrechterhaltung von *Legacy Systemen* entstehen

Eine Ausnahme von dem Prinzip, nur belegbare Kosten in die Betrachtung mit einzubeziehen, stellt die Empfehlung dar, die so ermittelten Kosten durch Kosten zu ergänzen, die aus einer ineffizienten Zusammenarbeit der Fachabteilungen mit der jeweiligen IT-Abteilung resultieren.

2.3.3 Ergebnisse verschiedener TCO Studien und Zusammenfassung

Entsprechend der oben beschriebenen Unterschiede kommen die verschiedenen Analysen auch zu deutlich unterschiedlichen Ergebnissen bei der Berechnung des TCO-Wertes. Eine Darstellung von verschiedenen Analysen für die TCO-Werte eines vernetzten Desktop-PCs liefert Abbildung 2-4.

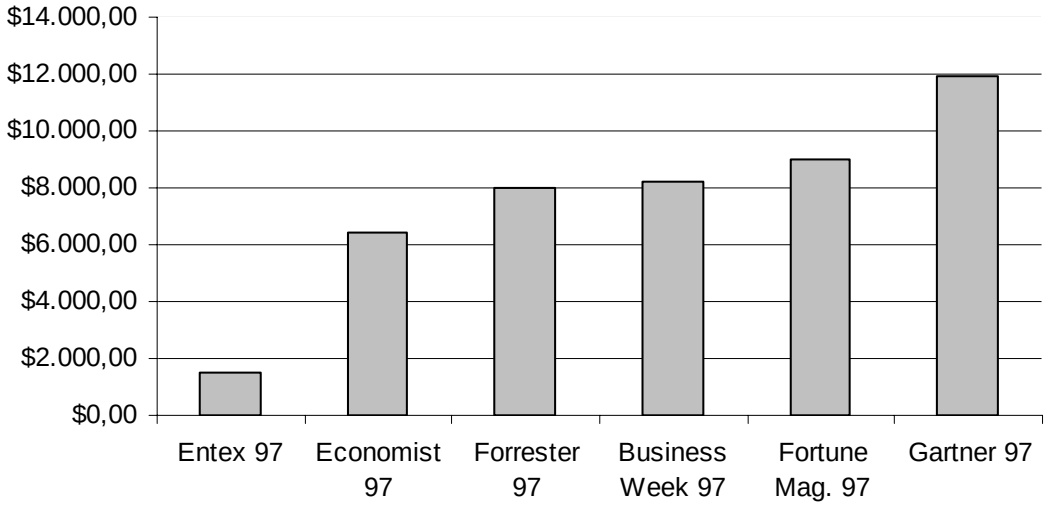


Abbildung 2-4 - Übersicht über TCO Ergebnisse verschiedener Research Gruppen[38]

Wie zu erwarten ergibt sich, entsprechend der Breite der Kostenkategorien der drei angesprochenen Modelle, der höchste Wert für das TCO-Modell der *Gartner Group*. *Forrester Research* liegt nur knapp darunter, während das (hier nicht dargestellte) *RCO* Modell der *META Group* durch die weitgehende Ausblendung indirekter Kosten zu einem sehr viel geringeren Wert führt.

Hieraus wird deutlich, dass aufgrund der fehlenden Standardisierung der TCO-Analyse ein Vergleich von TCO-Werten, die mit unterschiedlichen Modellen ermittelt werden, nicht sinnvoll ist. Dies gilt selbst, wenn ansonsten vergleichbare organisatorische Verhältnisse gegeben sind.

Riepl identifiziert trotz dieser Unterschiede vier allgemeine Bereiche, die Basiskostenfaktoren, in die sich die in den verschiedenen TCO-Modellen ermittelten Kosten einteilen lassen[39]. Diese stark vereinfachte Darstellung eines TCO-Konzeptes fasst deutlich die Ergebnisse der verschiedenen TCO-Analysen zusammen.

Basiskostenfaktoren:

- Wahrnehmung originärer Aufgaben einer EDV-Abteilung durch Endanwender (z.B. Peer to Peer Support) und Downtime

[38] *vgl. Groh 2000 S.43*

[39] *vgl. Riepl 1998 zitiert in Wild, Herges 2000 S.8*

- Vermögen an IT-Infrastrukturbestandteilen (Hard und Software, Übertragungsinfrastruktur)
- Technischer Support (z.B. Systemadministration)
- IT-bezogene Verwaltung (Verwaltung der EDV Abteilung / Schulungsmaßnahmen)

Abbildung 2-5 zeigt den Anteil der Basiskostenfaktoren an den gesamten TCO:

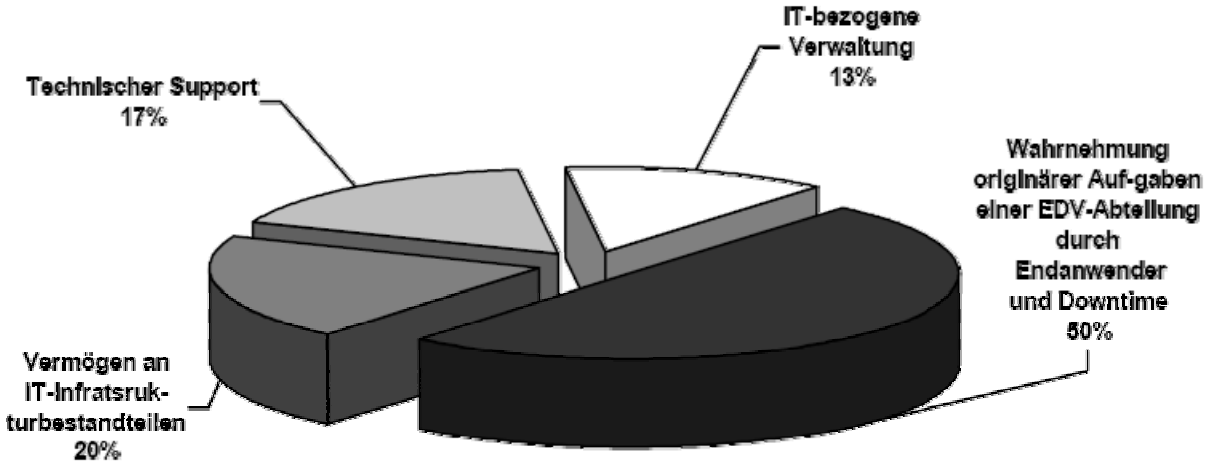


Abbildung 2-5 - Aufteilung von TCO auf ihre Basiskostenfaktoren[40]

2.4 Wissensintensive Arbeitsumgebungen und Wissenssammlungen

Vorbereitend auf die später folgende Identifikation von Benutzergruppen und die praktische Aufgabenstellung wird in diesem Abschnitt der Begriff der wissensintensiven Arbeitsumgebung präzisiert und eine Einführung in die Charakteristika von Wissenssammlungen gegeben. Neben der reinen Beschreibung von Wissenssammlungen wird der Begriff für die Zwecke dieser Arbeit abgegrenzt und eine Einordnung von Wissenssammlungen als ein Instrumentarium des Wissensmanagements gegeben.

2.4.1 Wissensintensive Arbeitsumgebungen

Der sich vollziehende "gesellschaftliche Strukturwandel hin zur Wissens- und Informationsgesellschaft"[41] manifestiert sich in der Arbeitswelt in der Anerkennung von

[40] *vgl. Riepl 1998 zitiert in Wild, Herges 2000 S.9*
[41] *vgl. Hartlieb 2002 S.1*

Wissen[42] ("intellektuelles Kapital"[43]) als vierten Produktionsfaktor, der ebenso wie die klassischen Produktionsfaktoren *Personal*, *Realkapital* und *Material*[44] geplant, gesteuert und überwacht werden muss. Für *Stimmler* ist das Wissen sowohl "Produktions- als auch Wettbewerbsfaktor eines Unternehmens"[45]. Hierdurch wird die Bedeutung des Wissens gerade für die Industrieländer hervorgehoben. Diese Hochkostenländer können in Zeiten der Globalisierung und des *Reverse Engineering*, in denen technologische Vorteile meist nur von kurzer Dauer sind, durch Wissen einen nachhaltigen Wettbewerbsvorteil schaffen. Die Nachhaltigkeit des Wissensvorsprunges ist durch die Eigenschaften des Wissens begründet. So nutzt sich Wissen, im Gegensatz zu materiellen Gütern, durch seine Benutzung nicht ab, sondern führt ganz im Gegenteil zu einer Vermehrung des Wissens und damit zum weiteren Ausbau des Wissens[46]. Aus diesem Grunde bezeichnet *Hartlieb*[47] das Wissensmanagement[48] für Hochpreisländer als "Königsweg internationaler Wettbewerbsfähigkeit".

Die wachsende Bedeutung des Wissens führt zu einer Zunahme des Anteils der Beschäftigten in wissensintensiven Arbeitsumgebungen. Selbst Autoren, die der Ansicht sind, dass die Entwicklung des Wissensbezugs der Arbeit überschätzt wird und dem aktuell zelebrierten Aufsehen um Knowledge-Management eher kritisch gegenüberstehen, gehen dennoch von einer moderat steigenden Bedeutung des Wissens im Arbeitsleben aus. So rechnet die konservative Schätzung von *Alvesson*[49] mit einer Steigerung des Anteils von Beschäftigten in wissensintensiven Arbeitsumgebungen von derzeit 10 bis 15% auf 15 bis 20% in ein bis zwei Jahrzehnten im Bereich Europa und Nordamerika[50].

Wissensintensive Arbeitsumgebungen treten nach *Alvesson*[51] in zwei Hauptkategorien von Organisationen auf: Zum einen im rein dienstleistungsorientierten Bereich der *Professional Service Firms (PSF)*. Hierzu zählen beispielsweise Kanzleien, Unternehmensberatungen,

---

[42] *für eine detailliert Definition von Wissen und den Zusammenhang zwischen Wissen, Information und Daten sei auf Davenport, Prusak 1998a S.27 ff. verwiesen*
[43] *vgl. Davenport, Prusak 1998a S.19*
[44] *vgl. Fraunhofer 2005*
[45] *vgl. Stimmler 2002 S.70*
[46] *vgl. Davenport, Prusak 1998a S.49*
[47] *vgl. Hartlieb 2002 S.1*
[48] *bezeichnet das Management von Wissenssystemen. Für eine detaillierte Vorstellung des organisatorischen Wissenssystems und der enthaltenen sozialen und technische Subsysteme sei auf Hartlieb 2002 S.58ff. verwiesen*
[49] *vgl. Alvesson 2004 S.7*
[50] *vgl. Alvesson 2004 S.9*
[51] *vgl. Alvesson 2004 S.18*

Werbeagenturen oder Investment-Banken. Die andere Kategorie, in der wissensintensive Arbeitsumgebungen verstärkt auftreten, umfasst Unternehmen, deren Erfolg zu erheblichen Teilen von Forschung und Entwicklung abhängt (*R&D firms*). Beispiele für diese Art von Unternehmen sind Pharma-, Biotechnologie-, oder andere Hoch-Technologie-Unternehmen. Charakteristisch für diese Unternehmen ist, dass die Entwicklungskosten eines Produktes ein Vielfaches der Herstellungskosten betragen. *Davenport und Prusak*[52] stellen in diesem Zusammenhang fest, dass die Grenzen zwischen Dienstleistungs- und Produktionsunternehmen durch intelligente Produkte zunehmend verschwimmen. So kann ein geliefertes Software-Paket sowohl als Produkt, als auch als Paket digital gelieferter Funktionen, also als Dienstleistung, angesehen werden.

Trotz der vielen unterschiedlichen Arten von wissensintensiver Arbeit, die in den verschiedensten Organisationen geleistet wird, existieren einige Eigenschaften mit deren Hilfe sich wissensintensive Arbeit von anderen Formen der Arbeit abgrenzen lässt. Die nachfolgende Abbildung 2-6 nimmt diese Abgrenzung auf der Ebene einzelner Prozesse vor.

[52] *vgl. Davenport, Prusak 1998a S.47*

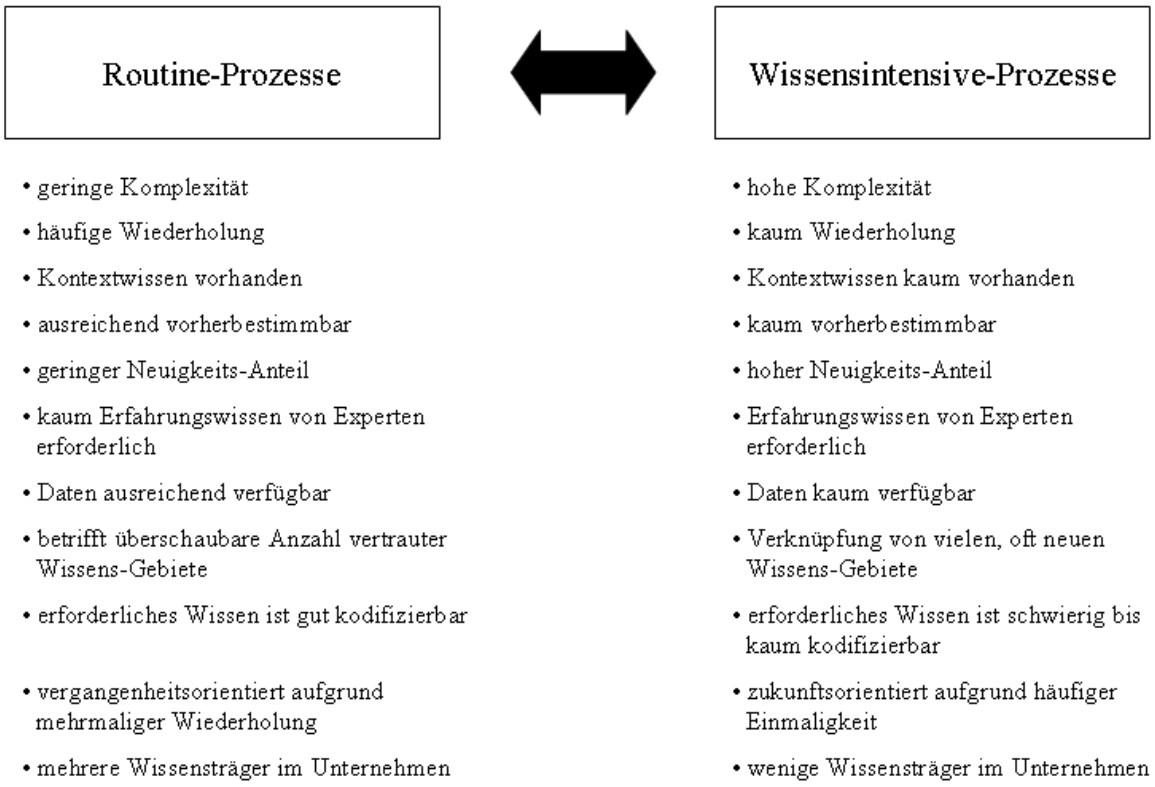


Abbildung 2-6 - Routine-Prozesse vs. Wissensintensive-Prozesse[53]

Charakteristisch für einen wissensintensiven Prozess sind demnach vor allem die Komplexität, die Einzigartigkeit und die daraus resultierende Unsicherheit der durchzuführenden Tätigkeit. Eine wissensintensive Arbeitsumgebung liegt vor, wenn der Tätigkeitsschwerpunkt eines Mitarbeiters aus wissensintensiven Prozessen besteht. Mitarbeiter, die in einer solchen Umgebung eingesetzt sind, werden als *Wissensarbeiter* (engl. *Knowledge Worker*) bezeichnet. Die Anforderungen an *Wissensarbeiter* lassen sich aufgrund der Vielseitigkeit der Arbeit nicht exakt definieren. Allerdings werden einige Eigenschaften in der Literatur genannt, durch die sich die Mehrheit der Wissensarbeiter auszeichnet. Die zwei wichtigsten Eigenschaften sollen hier kurz dargestellt werden.

Eine fundierte theoretische Ausbildung sehen viele Autoren[54] als eine der Schlüsselqualifikationen. Sie ermöglicht es dem *Wissensarbeiter*, theoretische Konzepte auf neue Situationen anzuwenden, steht im Ruf, die analytischen Fähigkeiten zu verbessern

[53] *vgl. Hartlieb 2002 S. 143*

[54] *vgl. beispielsweise Alvesson 2004 S.17, Davenport, Prusak 1998a S.214 oder Conklin 1996*

und dient als Nachweis für die wichtige Eigenschaft, selbständig zu lernen. Als weitere Eigenschaft wird die Teamfähigkeit gefordert, die es dem *Wissensarbeiter* ermöglicht, effizient mit verschiedenen Experten für bestimmte Bereiche zusammenzuarbeiten, und trotz unterschiedlicher Kenntnisse ein gemeinsames Verständnis herstellen zu können[55].

2.4.2 Wissenssammlungen

Der Begriff *Wissenssammlung* (engl. *Knowledge Repository*) wird in der Literatur für eine ganze Reihe unterschiedlicher Systeme und Einrichtungen verwendet. Während einige Autoren den Begriff der *Wissenssammlung* sehr weit fassen und beispielsweise Bibliotheken zu den *Wissenssammlungen* zählen, beschränken andere diesen auf elektronische Systeme wie beispielsweise Dokumenten-Management-, Groupware-, Intranet- oder Data-Warehouse-Systeme[56]. Auch das Internet wird häufig als *Wissenssammlung* bezeichnet[57]. Allerdings wird gerade in diesem Bereich sichtbar, dass die Relevanz von Wissen von höherer Bedeutung als die Vollständigkeit ist, da ein vorhandenes aber nicht auffindbares Wissen, nur Daten darstellt[58]. Die Auffindbarkeit des Wissens kann durch eine geeignete Strukturierung[59] und das Hinzufügen von Metadaten verbessert werden. Im Rahmen dieser Arbeit wird der Begriff *Wissensdatenbank (WDB)* benutzt, um eine *Wissenssammlung*, in der Informationen, in Form elektronischer Dokumente strukturiert und mit Metadaten versehen, abgelegt werden können, zu beschreiben.

Innerhalb des Wissensmanagements kann eine *Wissensdatenbank* dem technischen Subsystem[60] zugeordnet werden. Der Prozess der Generierung von neuen Informationen, die in der Datenbank hinterlegt werden, wird als Dokumentation bezeichnet. Der Prozess des Benutzens der Datenbank, um neues Wissen zu generieren wiederum wird als Information charakterisiert (vgl. Abbildung 2-7).

---

[55] *vgl. Conklin 1996*

[56] *vgl. Maedche 2002 S.428 ff.*

[57] *vgl. z.B. Schliwka 1998 S.85*

[58] *vgl. Davenport, Prusak 1998a S. 35*

[59] *zum Zielkonflikt der Strukturierung von Wissen vgl. Davenport, Prusak 1998a S. 146*

[60] *vgl. Hartlieb 2002 S.59*

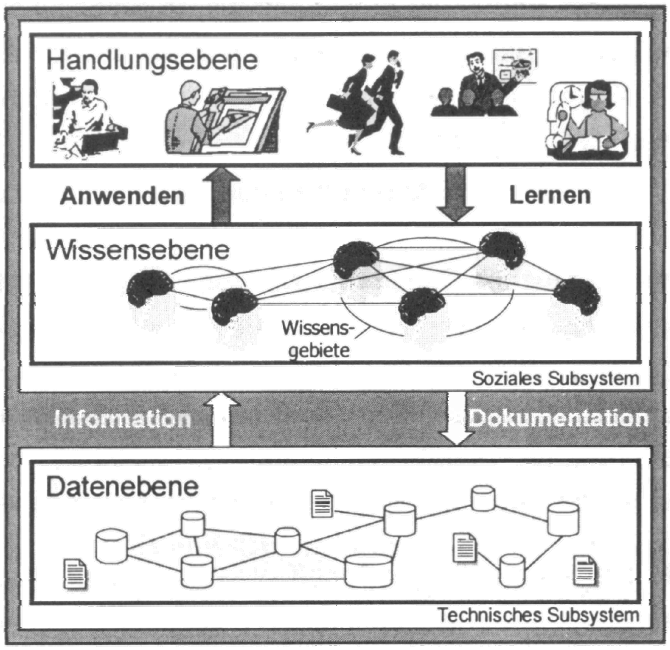


Abbildung 2-7 - Ebenenbetrachtung von Wissenssystemen[61]

Eine *Wissensdatenbank* kann das Wissensmanagement an mehreren Punkten sinnvoll unterstützen. Zunächst eignet sie sich zur *Externalisierung* von Wissen, also dazu, Wissen, das bisher personengebunden, d.h. nur in den Köpfen der Mitarbeiter, vorhanden war (implizites Wissen), zu dokumentieren, und somit beispielsweise in Form von Dokumenten sichtbar zu machen (explizites Wissen)[62]. Hierdurch wird die Abhängigkeit von einzelnen Experten unter den Mitarbeitern reduziert und die Voraussetzung für eine weitere Funktion, die eine *Wissensdatenbank* beim Wissensmanagement unterstützen kann, gelegt: Dem Wissenstransfer. Das in einer *Wissensdatenbank* abgelegte explizite Wissen kann, da es elektronisch vorliegt, sehr leicht transportiert werden und somit die Ausbreitung von Wissen innerhalb großer Organisationen, d.h. die Bildung eines kollektiven Wissens, erheblich beschleunigen. Darüber hinaus ermöglicht ein durch die Metadaten hergestellter Personenbezug, Wissensträger in einer Organisation ausfindig zu machen. Abbildung Abbildung 2-8 zeigt in Anlehnung an *Hartlieb* die beiden Dimensionen, in denen das

[61] *vgl. Hartlieb 2002 S.67*

[62] *Zur genauen Betrachtung der Transfer-Dimension des Wissens sei auf Hartlieb 2002 S.47ff. verwiesen. Die Grenzen der Externalisierbarkeit des Wissens werden bei Davenport, Prusak 1998a S. 150 anschaulich dargestellt*

Wissensmanagement durch eine *Wissensdatenbank* unterstützt werden kann. Die Helligkeit der einzelnen Bereiche steht hierbei für die Zugänglichkeit des Wissens.

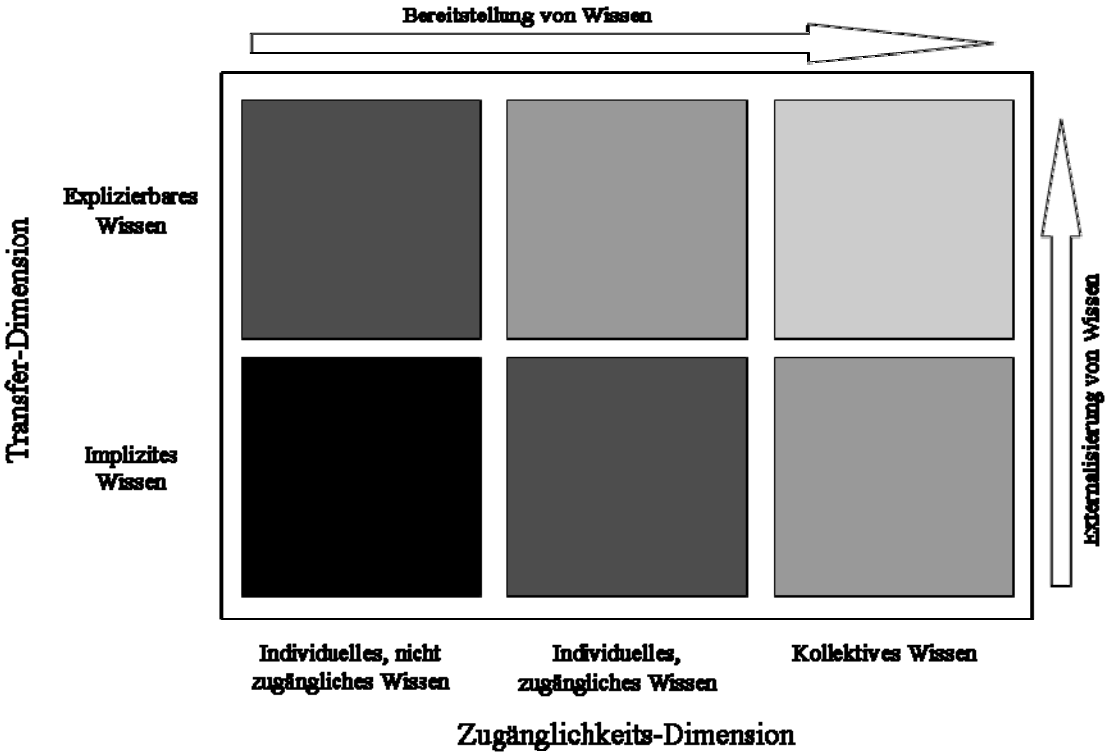


Abbildung 2-8 - Wissensdimensionen die durch Wissensdatenbanken unterstützt werden können[63]

Einschränkend sollte an dieser Stelle darauf hingewiesen werden, dass *Wissensdatenbanken*, wie oben erwähnt, nur ein Teil des technischen Aspektes des Wissensmanagements sind. Zwar weisen *Davenport und Prusak*[64] darauf hin, dass der angemessene Einsatz der technischen Infrastruktur ein Erfolgsfaktor eines erfolgreichen Wissensmanagements ist, doch wird die Wichtigkeit des sozialen Subsystems, z.B. die Schaffung von Wissensmärkten und einer offenen Unternehmenskultur, allgemein als eine ebenso wichtige Aufgabe gesehen[65].

[63] *in Anlehnung an Hartlieb 2002 S.49*

[64] *vgl. Davenport, Prusak 1998a S.295*

[65] *vgl. Alvesson 2004 S.138 ff. oder Davenport, Prusak 1998a S.251*

3 Bewertung von Clientkonzepten unter Berücksichtigung von Benutzergruppen eines spezifischen Anwendungsszenarios

Ziel dieses Kapitels ist es die verschiedenen Einflussfaktoren im Bereich der TCO und der Usability zu identifizieren, die für die Wahl der Client-Art im Anwendungsszenario von Wissensdatenbanken bestimmend sind. Zu diesem Zweck werden verschiedene Benutzergruppen für Wissensdatenbanken gebildet und die im vorherigen Kapitel vorgestellten Konzepte der Usability und der TCO kritisch gewürdigt sowie auf ihre Anwendbarkeit im konkreten Szenario untersucht. Anschließend erfolgt die Verknüpfung der unterschiedlichen Client-Arten mit den Konzepten der Usability und der TCO. Ebenso werden die identifizierten Benutzergruppen mit den einzelnen Aspekten der TCO und der Usability in Beziehung gesetzt um ein differenziertes Bild der Anforderungen der Benutzergruppen zu bekommen. Weiterhin wird im Folgenden mit den Rich Clients der zweiten Generation eine evolutionäre Client-Art vorgestellt, die geeignet ist die identifizierten Anforderungen besser zu erfüllen. Auf Basis der zuvor gewonnen Erkenntnisse wird schließlich eine Zuordnung von Benutzergruppen zu den nunmehr drei Client-Arten vorgenommen.

3.1 Identifikation von Benutzergruppen für Wissensdatenbanken

Hier werden für die oben beschriebenen Wissensdatenbanken Benutzergruppen gebildet, die es später ermöglichen, die unterschiedlichen Anforderungen im Hinblick auf Usability und TCO für diese Gruppen darzustellen und zu bewerten.

Davenport und Prusak[66] unterscheiden im Wissensmanagement vier verschiedene Rollen, von denen zwei auf den Einsatz von Wissensdatenbanken in wissensintensiven Arbeitsumgebungen angewendet werden können. Als größte interne Benutzergruppe sind die Wissensarbeiter zu nennen, die Wissensdatenbanken als Teil ihrer alltäglichen Arbeit operativ einsetzen. Dies beinhaltet sowohl die Informationssuche als auch die Ablage neuer Informationen. In wissensintensiven Arbeitsumgebungen sollte deshalb dieses operative Management von Wissen Bestandteil der Arbeit jedes Mitarbeiters sein, da nur durch die intensive Nutzung der Wissensdatenbank durch alle Mitarbeiter und der damit

[66] *vgl. Davenport, Prusak 1998a S.213ff.*

einhergehenden Relevanz und Aktualität der enthaltenen Informationen die Wissensdatenbank zu einem wertvollem Bestandteil im Wissensmanagement werden kann. Darüber hinaus existiert eine kleinere Benutzergruppe, nämlich die Mitarbeiter im Wissensmanagement selbst, deren Aufgabe die Strukturierung und Erweiterung der in einer Wissensdatenbank enthaltenen Informationen ist. Diese Tätigkeit entspricht zum einen den klassischen Administrationsaufgaben, wie die Bereitstellung der technischen Infrastruktur und der Benutzerverwaltung, beinhaltet auf der anderen Seite aber auch organisatorische sowie redaktionelle Tätigkeiten. So fällt die Ausarbeitung von Richtlinien für die Informationsablage genauso in das Arbeitsgebiet dieser Benutzergruppe, wie die Zusammenfassung und Neuordnung von vorhandenen Informationen.

Zusätzlich zum Einsatz im Wissensmanagement können Wissensdatenbanken jedoch auch zusätzlich als Grundlage für eine Veröffentlichung von Informationen genutzt werden, womit sich der Kreis potentieller Benutzer auch auf solche außerhalb der jeweiligen Organisation erweitert.

Neben dieser groben Dreiteilung der Benutzergruppen (Mitarbeiter im Wissensmanagement, Wissensarbeiter und externe Benutzer) existiert noch eine Fülle weiterer Kriterien nach denen eine Einteilung in Benutzergruppen vorgenommen werden kann. Tabelle 3-1 listet weitere Ordnungskriterien und ihre Ausprägungen auf.

Kriterien	**Ausprägungen**
Nutzungshäufigkeit	einmalig – täglich
Interaktionsmöglichkeiten:	lesend - schreibend – administrativ
Umstand des Zugriffs:	stationär / mobil(eigenes Gerät) / mobil(Fremdgerät)

Tabelle 3-1 - Ordnungskriterien zu Bildung von Benutzergruppen und deren Ausprägungen

Diese Kriterien überschneiden sich teilweise mit den bisher gebildeten Benutzergruppen. Im Folgenden werden die typischen Ausprägungen dieser Kriterien für die jeweiligen Benutzergruppen beschrieben und somit ein typisches Nutzungsszenario für die Benutzergruppen erstellt.

3.1.1 Externe Benutzer

Diese Benutzergruppe zeichnet sich dadurch aus, dass sie im Allgemeinen nur lesend auf die dargebotenen Informationen zugreifen kann, da eine Veränderung der in der Wissensdatenbank enthaltenen Informationen durch die Öffentlichkeit oft nicht gewünscht ist. Darüber hinaus ist im Allgemeinen auch der Zugriff auf die zu lesenden Informationen auf explizit freigegebene Informationsressourcen beschränkt. Die Nutzungshäufigkeit kann prinzipiell beliebige Ausprägungen annehmen und ist sehr von der Art der angebotenen Informationen abhängig. Jedoch ist zu erwarten, dass die Zahl der einmaligen Benutzer in dieser Gruppe besonders hoch sein wird. Entsprechend der Heterogenität dieser Benutzergruppe ist der Umstand des Zugriffs ebenso unterschiedlich.

3.1.2 Wissensarbeiter

Diese verfügen im Allgemeinen sowohl über Lese- als auch Schreibrechte in der Wissensdatenbank, wobei nicht ausgeschlossen wird, dass einige Informationen entsprechend der Hierarchiestufen der jeweiligen Organisation nicht zugreifbar sind. Die Nutzung ist in dieser Gruppe im Idealfall sehr hoch und wird meist über den eigenen (stationären) Arbeitsplatzrechner oder Laptop ausgeführt. Eine Untergruppe, die sich in den Nutzungsgewohnheiten unterscheidet, sind Mitarbeiter, die nur für einen begrenzten Zeitraum (beispielsweise im Rahmen eines Projektes) mit der Wissensdatenbank arbeiten. Sie nutzen die Wissensdatenbank nur über diesen begrenzten Zeitraum regelmäßig, wobei der Zugriff meist über Fremdgeräte erfolgt. Beispiele hierfür sind externe Berater oder, für die Dauer eines Projektes, hinzugezogene Mitarbeiter aus Fachabteilungen.

3.1.3 Mitarbeiter im Wissensmanagement

Neben Lese- und Schreibrechten besitzen Mitarbeiter im Wissensmanagement auch administrative Rechte in der Wissensdatenbank. Die Nutzungshäufigkeit der Wissensdatenbank ist auf Grund der vielfältigen Aufgaben dieser Benutzergruppe ebenfalls hoch. Der Zugriff erfolgt in der Masse der Fälle über einen stationären Computer am Arbeitsplatz.

3.2 Kritik der vorgestellten Konzepte zur Bewertung der Vorteilhaftigkeit von IT-Investitionen

Die im zweiten Kapitel vorgestellten Verfahren zur Bewertung von IT-Investitionen, zum einen die Kosten-Nutzen-Rechnung für Usability-Maßnahmen und zum anderen das TCO-Modell, sollen nachfolgend kritisch gewürdigt werden. Während bei der Kosten-Nutzen Rechnung für Usability-Maßnahmen der Schwerpunkt auf der Verbesserung von Software und der Bewertung des damit einhergehenden Nutzengewinns liegt, untersucht das TCO Modell im Gegensatz dazu die Kostenstruktur einer IT-Infrastruktur und versucht, durch eine detaillierte Klassifikation der Kosten, Einsparpotentiale aufzuzeigen. Zu jedem dieser Verfahren werden die häufigsten in der Literatur anzutreffenden Kritikpunkte dargestellt.

3.2.1 Kosten-Nutzen Rechnungen für Usability

Die Kritikpunkte zu Kosten-Nutzen Rechnungen für Usability-Maßnahmen lassen sich grob in zwei Kategorien einteilen. Auf der einen Seite wird die Art und die Rechtfertigung der Schätzungen, die den Nutzenzuwachs durch Usability bestimmen, kritisiert und auf der anderen Seite die unzureichende Eingliederung in praktische, betriebswirtschaftliche Problemstellungen.

Die Schätzung des Nutzengewinns basiert auf der Extrapolation der Ergebnisse verschiedener Studien, die jeweils zu einer gesamten (meist zeitlichen) Ersparnis addiert werden. Allerdings spiegeln diese, unter konstanten Rahmenbedingungen gewonnenen Ergebnisse die in der Realität erzielbaren Ergebnisse nur unzureichend wider. Darüber hinaus führt *Rosenberg*[67] an, dass die empirische Grundlage, auf Grund welcher die Nutzenschätzungen durchgeführt werden, sehr dünn ist. und konstatiert ein Fehlen neuerer Studien, die als Rechtfertigung für diese Schätzungen dienen könnten. Er führt weiterhin aus, dass die quantitativen Nutzenschätzungen meist lediglich durch qualitative Aussagen des Wirkens einer verbesserten Usability fundiert werden. Ein weiteres Problem, das bei der Schätzung auftritt, ist, dass jeweils eine Nutzendifferenz ermittelt werden muss. Das bedeutet, dass nicht ein absoluter Nutzen geschätzt wird, sondern die Verbesserung des Nutzens, die auf ein bestimmtes Usability-Programm zurückzuführen ist. Dies macht es erforderlich, (zumindest gedanklich) zwei Schätzungen abzugeben. Zunächst ist für das zu realisierende System ohne Durchführung eines bestimmten Usability-Programms der

[67] *vgl. Rosenberg 2004 S.24f.*

Nutzen zu schätzen, und dann die Schätzung unter Einbeziehung des Usability-Programms zu wiederholen. Hierdurch ergibt sich zum einen eine hohe Unsicherheit, zum anderen ist die Überprüfung des Erfolgs eines durchgeführten Programms schwierig, da Vergleichswerte, die den Nutzen des Systems ohne Usability-Programm wiedergeben, nicht verfügbar sind. Für eine Studie über die Wirkung eines Usability-Programms müsste also die gleiche Ausgangssituation zweimal vorliegen, um die Wirkung eines solchen Programms isolieren zu können. Eine solche Situation ergibt sich in der Realität praktisch nie, und auch rein zum Zwecke einer Studie stellt die Herstellung einer solchen Ausgangssituation ein kaum zu überwindendes Hindernis dar. Die Isolation des durch Usability-Maßnahmen erreichten Umsatzzuwachses ist auf Grund der multikausalen Einflüsse ebenfalls nahezu unmöglich[68].

Die unzureichende Beachtung betriebswirtschaftlicher Rahmenbedingungen bei der Kosten-Nutzen Rechnung von Usability wird an mehreren Stellen deutlich. Zunächst zeigt die Berechnung der Ersparnisse mit Hilfe der Mitarbeiterstundensätze, dass die Personalkosten als völlig variabel angenommen werden[69]. Der berechnete Wert entspricht in der Praxis, wenn man die oben erwähnten Unwägbarkeiten der Schätzung hier nicht beachtet, zunächst einmal dem maximalen Einsparpotential[70]. Ausgehend von diesem Wert müssen die realisierbaren Einsparungen differenzierter betrachtet werden. So treten die skizzierten Einsparungen in Bereichen der "informationstechnischen Massenproduktion", wie beispielsweise der Annahme von Bestellungen in einem Call-Center, mit hoher Wahrscheinlichkeit eher zu Tage. Sie können entweder durch Nutzung der frei werdenden Kapazitäten (Übernahme neuer Aufträge) oder durch Einsparungen im Personalaufwand (Entlassung von Mitarbeitern) im konkreten Fall realisiert werden. Dieses Beispiel macht deutlich, dass eine verbesserte Usability nicht alleine Einsparungen garantiert, sondern zusätzliche organisatorische Maßnahmen zur Nutzung des Potentials begleitend erforderlich sind. Usability-Vorteile in Bereichen, die durch eine hohe Heterogenität des Arbeitsumfeldes gekennzeichnet sind, (insbesondere also die im Abschnitt 2.4 beschriebene Arbeitsumgebung) hingegen deutlich schwerer monetär zu realisieren. Dies findet seine Begründung darin, dass auf Grund einer fehlenden eindimensionalen Messgröße für den Erfolg der Arbeit, sich die direkten Vorteile kaum quantitativ, sondern

---

[68] *vgl. Rosenberg 2004 S.26f.*

[69] *ein Beispiel für eine auf solch einseitigen Annahmen basierende Rechnung liefert Hirschmeier 2002 S.58f. siehe Anhang (b)*

[70] *vgl. hierzu das Problem der Sichtbarkeit von Einsparungen Nielsen 1993 S.3ff.*

eher qualitativ zeigen (z.B. bessere Arbeitszufriedenheit/-qualität) und so nur indirekt auf monetäre Größen wirken.

Eine nicht ausreichende Beachtung von betriebswirtschaftlichen Rahmenbedingungen zeigt sich auch in der von *Mayhew und Mantei*[71] vorgeschlagenen Herangehensweise an die Kosten-Nutzen-Rechnung eines Usability-Programms. Ziel dieses Verfahrens ist es, einem Usability-Ingenieur wirtschaftliche Argumente für die Etablierung eines Usability-Programms bei der Entwicklung von Software an die Hand zu geben. Das vorgeschlagene Verfahren empfiehlt, mit einer maximalen Maßnahme zu beginnen und diese, wenn die Berechnung einen deutlich positiven Wert ergibt, durchzuführen. Erst bei einem negativen Ergebnis einer solchen Berechnung soll die Berechnung iterativ mit einem schlankeren Usability-Programm wiederholt werden, bis sich ein positiver Wert ergibt. Es tendiert demnach dazu, möglichst umfassende Usability-Programme durchzuführen, und sucht deshalb nicht zwangsläufig nach einem Kosten-Nutzen optimalen Usability-Programm. Die Begründung für dieses Vorgehen liegt vermutlich zum einen darin, dass die Anzahl der Schätzungen und der damit verbundene Aufwand möglichst gering gehalten werden soll und zum anderen in dem Irrglauben, dass ein lediglich vorteilhaftes Usability-Programm seine Durchführung implizit rechtfertigt, und somit die Suche nach einem optimalen Programm die Durchführungschancen nicht weiter erhöht. Hier wird die isolierte Betrachtungsweise der Berechnung deutlich. In der Realität sind die verfügbaren Ressourcen beschränkt und eine Investition in Usability konkurriert mit anderen ergreifbaren Maßnahmen. In einem solchen Szenario werden nicht alle positiven, sondern nur die besten verfügbaren Investitionen durchgeführt. Eine Optimierung kann demnach in der Realität die Chancen für ein Usability-Programm sehr wohl erhöhen.

3.2.2 Konventionelle TCO Modelle

Obwohl TCO-Modelle im Vergleich zu klassischen, betriebswirtschaftlichen Kostenrechnungsverfahren bei der Bewertung einer IT-Infrastruktur eine erheblich bessere Darstellung der Realität ermöglichen, und somit "nicht unerheblich zum Treffen realitätskonformer Entscheidungen"[72] beitragen, werden in der Literatur zahlreiche Kritikpunkte an TCO-Modellen angeführt. Diese gliedern sich in potentielle Fehler, welche

[71] *vgl. Mayhew ,Mantei 1994 S.36 ff.*

[72] *vgl. Wild, Herges 2000 S.27*

bei der Erhebung der TCO gemacht werden können, und konzeptionelle Kritik, die auf die Grundkonstruktion der TCO zielt und darauf aufbauend Verbesserungsvorschläge anbietet.

Schon bei der Vorstellung der einzelnen TCO-Modelle hat sich gezeigt, dass die Bewertung und Abgrenzung der indirekten ("nicht budgetierten") Kosten einer der strittigsten Punkte bei der Erhebung der TCO sind. *Hirschmeier*[73] schätzt, dass ca. 30 % der TCO auf diese intransparenten Kostenarten entfällt, und sich hierdurch entsprechende Unterschiede in den Analysen ergeben können. Als Lösungsansatz für dieses Problem wird eine stärkere Ausrichtung der Kostenerfassung an Prozessen mit Hilfe der Prozesskostenrechnung vorgeschlagen, um die Transparenz in diesem Bereich zu erhöhen. Weiterhin wird die Prozesskostenrechnung ebenso als Mittel vorgeschlagen, den statischen TCO-Wert zur besseren Abbildung der einzelnen Lifecycle-Phasen einer IT-Infrastruktur (z.B. Entwicklung, Einführung, Betrieb) zu dynamisieren, da die anfallenden Kosten nicht gleichmäßig aus dem Besitz, sondern in einzelnen Phasen in unterschiedlicher Höhe aus dem Betrieb einer IT-Infrastruktur resultieren[74]. So entstehen beispielsweise bei der Einführung und im ersten Zeitraum des Betriebs einer neuen IT-Infrastruktur, bedingt durch fehlende Erfahrung mit dem System, typischerweise höhere Kosten als in den darauf folgenden Jahren.

Die zeitliche Vergleichbarkeit von TCO-Werten ist auf Grund der Verwendung von Anschaffungskosten (im Vergleich zu Wiederbeschaffungskosten) und dem mit der schnellen technologischen Entwicklung einhergehendem Preisverfall stark eingeschränkt. Die technologische Entwicklung erfordert darüber hinaus eine stetige Anpassung des TCO-Modells, was die zeitliche Vergleichbarkeit zusätzlich beeinträchtigt. Schließlich kritisiert *Lütge*[75] noch, dass der verwendete Begriff TCO zu unrecht suggeriert, dass alle Kosten berücksichtigt werden und weist richtigerweise darauf hin, dass es sich auch bei den TCO lediglich um eine Auswahl von mit der IT-Infrastruktur in Zusammenhang stehenden Kosten handelt. Wie bereits erwähnt, finden zum Beispiel die in anderen Kostenrechnungsverfahren regelmäßig anzutreffenden kalkulatorischen Kosten keinen Eingang in das TCO Modell.

Neben den oben genannten Kritikpunkten, die spezielle Aspekte der TCO-Analyse kritisieren, finden sich in der Literatur zwei Hauptkritikpunkte die das gesamte TCO-

---

[73] *vgl. Hirschmeier 2002 S.22*

[74] *vgl. Karner 2005 S.11 oder Lütge 2002 S.16*

[75] *vgl. Lütge 2002 S.17*

Modell betreffen. Die mangelnde Standardisierung des Verfahrens wird übereinstimmend als einer der größten Nachteile gesehen. Sie ist zum einen dafür verantwortlich, dass viele unterschiedliche und deswegen zueinander nicht kompatible Modelle existieren - *Lütge*[76] spricht von einem Vergleich von Äpfeln mit Birnen - und hat es zum anderen ermöglicht, dass TCO-Modelle durch Anpassung an die Bedürfnisse eines spezifischen Herstellers teilweise zu reinen Marketinginstrumenten verkommen sind[77].

Der zweite Hauptkritikpunkt betrifft die ausschließliche Betrachtung der durch die IT-Infrastruktur entstehenden Kosten ohne diesen Kosten einen durch die IT-Infrastruktur bedingten Wertzufluss gegenüberzustellen. Hier besteht ein Zielkonflikt. Während die übrigen Geschäftsbereiche kosten- und leistungsorientiert geführt werden, kommt es bei alleiniger Anwendung des TCO-Modells im IT-Bereich nur zu einer rein kostenorientierten Steuerung. Diese Beschränkung kann langfristig zu Wettbewerbsnachteilen führen, wenn notwendige aber teure Investitionen in die IT-Infrastruktur aus Kostengründen nicht getätigt werden. *Hirschmeier*[78] überzeichnet dieses Innovationsrisiko und behauptet, dass bei reiner TCO-Ausrichtung in den Unternehmen immer noch Terminals eingesetzt würden, da ihre Kosten deutlich unter denen aller heute verwendeten Clienttechnologien liegen. Auf Grund dieser "wertverzehrbetonten"[79] Sichtweise eignet sich das TCO-Modell nicht als alleiniges Steuerinstrument für die IT-Infrastruktur[80], sondern es müssen der TCO-Analyse zusätzliche Instrumentarien zur Seite gestellt werden. Auf diese Problematik weist auch die *Gartner Group* ausdrücklich hin[81].

Eine diesbezügliche Verbesserung soll durch die Einbeziehung von, durch die IT-Infrastruktur bedingten Wertezuflüssen erreicht werden. Diese Einbeziehung ist aber gerade bei einer IT-Infrastruktur problematisch, da viele Nutzenpotentiale innovativer Technologien in der Zukunft liegen[82]. Unter anderem aus diesem Grund mangelt es derzeit an theoretischen Konzepten, welche die Leistungen einer IT-Infrastruktur (TBO, Total Benefit of Ownership) in einer Weise darstellen, die mit dem TCO verglichen werden kann[83]. Die Kombination aus TCO und TBO ergibt dann ein Modell, mit dem die

---

[76] *vgl. Lütge 2002 S.17*

[77] *vgl. Potthoff 1998 S.8, Wild,Herges 2000 S.31 oder Karner 2005 S.11*

[78] *vgl. Hirschmeier 2002 S.22*

[79] *vgl. Wild, Herges 2000 S.27*

[80] *vgl. Li2003 S.5f.*

[81] *vgl Biskamp 1998 S.10*

[82] *vgl. Hirschmeier 2002 S.12*

[83] *vgl. Wild, Herges 2000 S.28*

Wertstruktur eine IT-Infrastruktur realitätsnah abgebildet werden kann. Beispiele für solche Modelle sind das TVO–Modell (Total Value of Ownership) von IDC oder auch das unten dargestellte TEI–Modell (Total Economic Impact) der *Giga Information Group*. Bei letzterem Modell tauchen die TCO als Faktor neben dem oben angesprochenen Faktor Nutzen auf. Ergänzt werden diese beiden Faktoren durch die Bewertung der Flexibilität der IT-Infrastruktur die es ermöglicht, die Einflüsse zukünftiger Entwicklungen auf die bestehende Infrastruktur im Modell zu bewerten. Abschließend erfolgt die Berücksichtigung von Risiken für jeden der drei Faktoren (vgl. Abbildung 3-1).

Abbildung 3-1 - Übersicht: Total Economic Impact (TEI) [84]

3.3 Gegenüberstellung von Thick-Client und Thin-Client

Im Grundlagenkapitel 2.1 wurden die verschiedenen Client-Arten und ihre, im Rahmen dieser Arbeit verwendete, Interpretation vorgestellt. Im nachfolgenden Abschnitt soll nun der Bezug zu den ebenfalls im 2. Kapitel vorgestellten Themenbereichen Usability und TCO hergestellt werden. Anhand von Szenarien und praktischen Beispielen wird dargestellt, warum die Vorteile eines browser-basierten Thin-Clients eher im Bereich der TCO liegen und umgekehrt die Vorteile eines Thick-Clients im Bereich der Usability zu suchen sind.

3.3.1 Usability

Im Abschnitt 2.2.2 wurden einige Design Grundsätze vorgestellt, die hier als Strukturierung dienen sollen, um die spezifische Eignung eines Thick-Clients im Vergleich zu einem browser-basierten Thin-Client darzustellen.

---

[84] *vgl. Wild, Herges 2000 S.30*

Die geforderte *Consistency*[85] kann durch die Verwendung von Thick-Clients, vor allem wenn es um die Einhaltung externer, also programmübergreifender, Konsistenz geht, einfacher erreicht werden. Da der Client in den meisten Fällen ein innerhalb des Betriebssystems ablaufendes Programm ist, benutzt er zur Darstellung automatisch die von diesem bereitgestellten Ressourcen wie zum Beispiel Widgets. Auch ist es im Vergleich zum Web-Browser einfacher möglich, die für Programme geltenden Konventionen einzuhalten (z.B. das Vorhandensein und der Aufbau einer Menüstruktur oder das kontrollierte Beenden einer Applikation).

Eine Web-Anwendung ist aus mehreren Gründen, unter anderem auf Grund von Sicherheitserwägungen, im Zugriff auf die zu Verfügung stehende Betriebssystem-ressourcen beschränkt. Dies führt dazu, dass die Möglichkeiten einer Web-Anwendung, mit dem Benutzer zu interagieren, im Bereich des Outputs auf die im Web-Browser angezeigte Seite, und beim Input auf Maus und Tastatur beschränkt sind. Hierdurch sind die Wahlmöglichkeiten zur Einhaltung des Design Grundsatzes *Consideration of User Resources* eingeschränkt. Beispielsweise ist die mit einem Thick-Client implementierbare Audio-Signalisierung die auf einen veränderten Zustand der Anwendung hinweist (z.B. Benachrichtigung über neue E-Mails oder Aufgaben) mit einer web-basierten Anwendung nur schwer möglich.

Eine der größten Schwachstellen von web-basierten Anwendungen ist das mangelnde bzw. verzögerte *Feedback*, das bedingt ist durch die notwendige Kommunikation mit dem Server und dem damit verbundenen Neuaufbau der Seite. Sie führt zum einen dazu, dass das Verhältnis zwischen Denken, Durchführen und Warten bei Nutzung einer Web-Anwendung sehr unausgeglichen ist[86]. Zum anderen können diese Verzögerungen zur Folge haben, dass in Thick-Clients erfolgreich verwendete Metapher (z.B. das Konzept von Registerkarten) durch die langsame Reaktion entwertet werden[87].

Dem Grundprinzip der *Error Prevention and Recovery* kann in Web-Anwendungen ebenfalls nur eingeschränkt bzw. mit erheblichem Aufwand entsprochen werden. Während das Problem der Validierung von Benutzereingaben, z.B. durch den Einsatz von *JavaScript* (als Alternative zur server-seitigen Validierung, vgl. o.g. *Feedback*-Problem), noch mit einigem Aufwand gelöst werden kann, ist die Bereitstellung von, in Thick-Client

[85] *vgl. Nielsen 1993 S.227f. zur Bedeutung der Consistency für die Produktivität der Benutzer*
[86] *vgl. Spolsky 2001 S.122ff.*
[87] *vgl. Spolsky 2001 S.40*

Applikationen üblichen, Undo-Funktionalitäten in Web-Anwendungen die absolute Ausnahme. Eine weitere Technik, die in Thick-Client Applikationen zum Einsatz kommt, ist die Nutzung von Dialogfenstern (z.B. zum Abfragen ob Änderungen übernommen werden sollen). Durch ihren Einsatz behält der Benutzer visuell den Bezug zum Kontext des Hauptfensters, wird aber gleichzeitig angeleitet, zunächst das Dialogfenster zu bearbeiten. In Web-Anwendungen wird ein solches Szenario meist durch Aufruf einer neuen Seite gelöst (Kontextverlust) oder die entsprechende Seite wird, in Anlehnung an das Konzept einer Dialogbox, in einem neuen (oft verkleinertem) Browserfenster geöffnet (vgl. Abbildung 3-2).

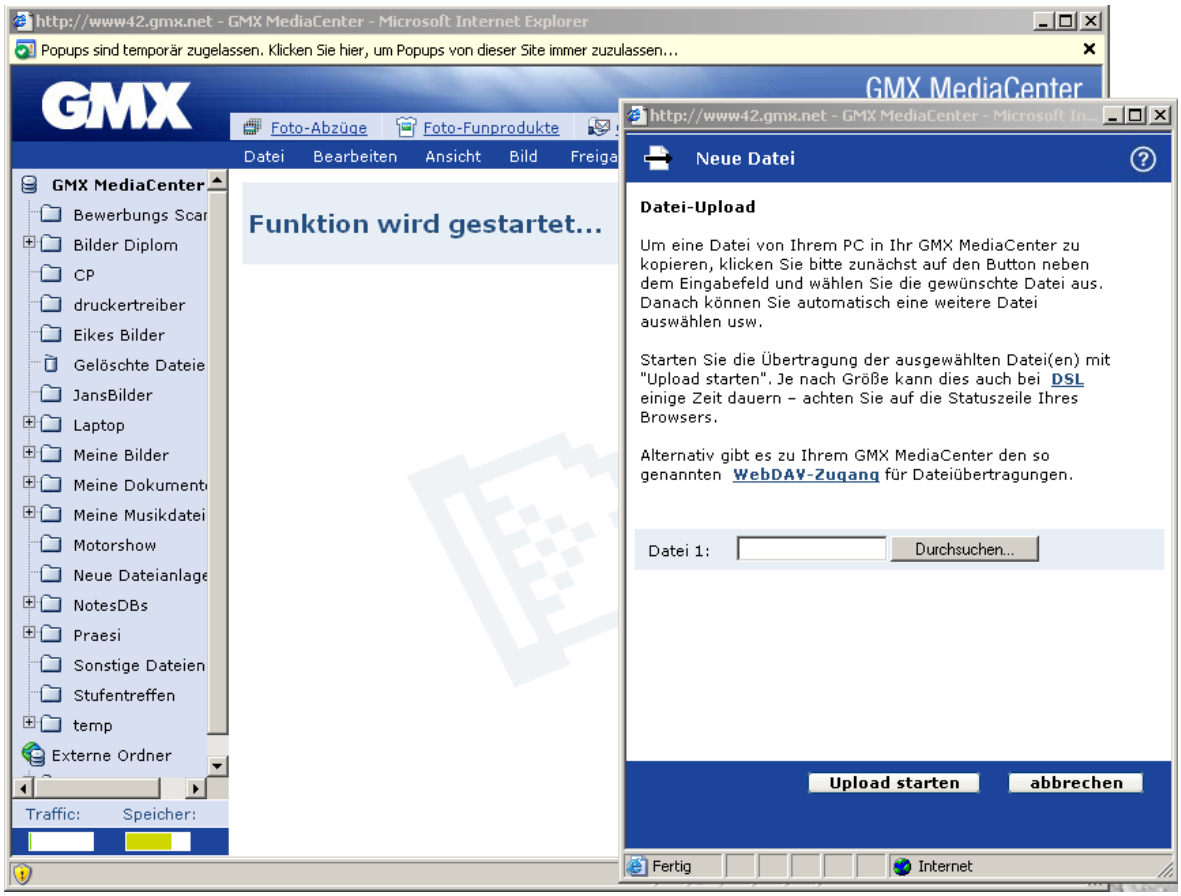


Abbildung 3-2 - Simulation der Dialogbox-Funktionalität in Web-Anwendungen

Allerdings fehlt in diesem Fall die entsprechende Sperrung des Hauptfensters, da der Browser den Zusammenhang zwischen beiden Fenstern nicht kennt. Darüber hinaus führt ein solches Vorgehen oft zu Problemen mit so genannten „Pop-Up-Blockern“, da diese das zu öffnende Dialogfenster nicht von einer gewöhnlichen Werbeeinblendung unterscheiden können. Als letztes Beispiel für die mangelnde Vermeidung von Fehlern sei noch auf den möglichen Verlust von Eingaben beim Schließen des Browserfensters oder bei Aufruf

einer neuen Seite in der Adresszeile hingewiesen, da ein, in Thick-Client Anwendungen üblicher Warnhinweis, technisch nicht sinnvoll realisierbar ist.

Dem Design-Grundsatz der *User Control* kann in Web-Anwendungen aus technischen Gründen in vielen Fällen ebenfalls nur bedingt Rechnung getragen werden. Der Einsatz von Tastaturkürzeln, die individuelle Anpassung der Benutzeroberfläche und die Anpassung der Oberfläche auf verschiedene Auflösungen sind Problemstellungen, die in Web-Anwendungen nur unzureichend adressiert werden können. Auch der Kontrolle des Benutzers über das Tempo der Interaktion sind durch mögliche zeitbedingte Sitzungsverluste Grenzen gesetzt. Ein Anwendungsbeispiel hierfür ist die Wiederaufnahme eines Vorgangs nach einem längeren Telefongespräch oder der Mittagspause.

3.3.2 Total Cost of Ownership (TCO)

Trotz der oben angeführten Nachteile browser-basierter Thin-Clients, werden aktuell über 70% aller neuen Anwendungsprojekte browser-basiert realisiert[88]. Die Gründe für diese Dominanz liegen in Kostenvorteilen die mit dem Einsatz von browser-basierten Thin-Clients einhergehen. In diesem Abschnitt soll anhand der Kostenfaktoren des TCO-Modells eine Einordnung dieser Einsparpotentiale gegeben werden.

Bei den direkten Kosten ermöglicht der Einsatz von Thin-Clients die Realisierung von Einsparungen zum einen im Bereich der *Hard- und Software* und zum anderen im Bereich *Operations* und hier insbesondere beim *Technischen Support*.

Kostenvorteile ergeben sich einerseits im Bereich der *Hard- und Software*, die für die eigentlichen Geschäftsprozesse des Unternehmens verantwortlich ist. Zwar ist davon auszugehen, dass auf Grund der funktionalen Fokussierung auf den Server die Kosten für Server-Hardware beim Einsatz von Thin-Clients im Vergleich zu Thick-Clients steigen werden. Dieser Effekt wird jedoch durch Einsparungen auf der Client-Seite überkompensiert. So ist die client-seitig benötigte Rechenleistung beim Einsatz von Thin-Clients grundsätzlich geringer als bei Thick-Clients. Somit kann bei den Thin-Clients leistungsschwächere und damit kostengünstigere Hardware eingesetzt werden, die zusätzlich oft auch länger genutzt werden kann. Bei Einsatz einer entsprechend konfigurierter Thin-Client Hardware können darüber hinaus noch Einsparungen im Bereich der Client-Software erzielt werden (z.B. Anwendungssoftware oder auch das

[88] *vgl. Miedl 2003*

Betriebssystem). Ein weiterer Kostenvorteil einer Thin-Client Umgebung kann dem Bereich der Software für die eigentliche EDV-Abteilung zugeordnet werden. Gemeint ist der durch Thin-Clients reduzierte Bedarf an Systemmanagement Software, da Programme, die eine zentrale Administration eingesetzter Thick-Client Anwendungen ermöglichen, nicht erforderlich sind.

In der TCO-Kostenkategorie *Operations* sind Einsparungen insbesondere in der Unterkategorie *Technischer Support* zu suchen. Der Einsatz von Thin-Clients reduziert in diesem Bereich Kosten für Software-Installationen, Applikations-Management sowie Sicherung und Archivierung. Ermöglicht wird dies durch die zentrale Administration einer web-basierten Thin-Client Umgebung. So erfordert beispielsweise die Installation einer neuen Version einer Anwendung keinerlei Eingriff auf der Client-Seite. Weiterhin sind Einsparungen bei Problemlösungen und dem Erhalt der Systemstabilität zu sehen. Da die Clients zum Beispiel bei Updates unverändert bleiben, sind Wechselwirkungen, die auf den Clients durch die Installation von verschiedenen (Thick-Client-)Anwendungen auftreten können[89], nahezu ausgeschlossen. Selbst beim Einsatz von Thick-Client-Hardware kann durch eine einheitliche, schlanke Softwareinstallation (z.B. Betriebssystem und Browser) der Aufwand für eine Wiederherstellung oder die Bereitstellung eines Arbeitsplatzes minimiert werden.

Bei den indirekten Kosten entstehen sowohl im Bereich der *Downtime* als auch bei den *End User Operations* durch den Einsatz einer Thin-Client Umgebung Einsparpotentiale. Da das Update eines Anwendungssystems lediglich auf dem Server erfolgt, können geplante Ausfallzeiten im Vergleich zu einem dezentralen Thick-Client Update minimiert werden. Ungeplante Ausfallzeiten der Clients werden durch ihre geringere Komplexität (vgl. die oben angesprochene schlanke Thin-Client Installation) unwahrscheinlicher und schneller behebbar. Dem entgegen steht allerdings das Risiko eines Serverausfalls, der in einer Thin-Client Infrastruktur auf Grund der nicht vorhandenen Offline-Fähigkeiten der Clients deutlich schwerer wiegt als in einer Thick-Client Infrastruktur. Allerdings gilt auch hier analog zur oben angeführten Hardware-Diskussion, dass der für den Server zusätzlich zu betreibende Aufwand nur ein Bruchteil der Kosten erfordert, die für Thick-Clients aufgewendet werden müssen.

---

[89] *Bei Microsoft Windows BS sind die beschriebenen Probleme unter dem Stichwort DLL-Hell bekannt*

Im Bereich der *End User Operations* ergeben sich die Einsparpotentiale aus der im vorherigen Abschnitt bemängelten Einfachheit der Oberfläche eines browser-basierten Thin-Clients. So entfallen hier zumeist indirekte Kosten, die durch unproduktiven Umgang mit der Oberfläche (Kostenkategorie *Futzing*) oder durch Entwicklung eigener Software entstehen. Zu diesen Einsparungen muss jedoch gesagt werden, dass sie zum einen weitestgehend von der Thematik der Clients unabhängig sind (die Höhe der Futzing-Kosten ist beispielsweise auch von den organisatorischen Rahmenbedingungen abhängig), und zum anderen werden diese Einsparungen (z.B. keine Entwicklung eigener Makros/Software durch Mitarbeiter) nur rein kostenmäßig betrachtet. Ein etwaiger Nutzenentgang durch das Fehlen dieser Möglichkeit wird nicht in diese Berechnung einbezogen.

3.4 Zusammenhänge zwischen Usability und TCO

Im vorhergehenden Abschnitt wurde gezeigt, inwieweit die Wahl des Clients (web-basierter Thin-Client gegenüber einem Thick-Client) Auswirkungen auf die erreichbare Usability bzw. die erzielbaren TCO für ein Gesamtsystem hat. Zwar sind diese beiden Größen außer von der Wahl des Clients auch von vielen anderen Faktoren abhängig, doch gibt die Auswahl der Client-Art dennoch technische Rahmenbedingungen vor, die die minimal erzielbaren TCO bzw. die maximal realisierbare Usability determinieren. In diesem Abschnitt sollen, nach einer kurzen generellen Betrachtung des Verhältnisses von Usability und TCO, die im Abschnitt 3.1 definierten Benutzergruppen in die Betrachtung miteinbezogen werden, in dem die Bedeutung der TCO und verschiedener Usability-Komponenten für die Benutzergruppen dargestellt wird.

3.4.1 Generelles Verhältnis zwischen Usability und TCO

Konventionelle TCO-Modelle betrachten lediglich die Kosten einer IT-Infrastruktur[90]. Das bedeutet, dass Usability–Aspekte, die typischerweise entweder durch zusätzlichen Entwicklungsaufwand oder höhere Softwarepreise die Kosten steigern, in konventionellen TCO-Modellen nicht berücksichtigt werden. Auf der anderen Seite implizieren die zur Berechnung von Usability-Maßnahmen vorgeschlagenen Ansätze (vgl. Abschnitt 2.2.3) mit ihrer Anweisung die Berechnungen mit einem maximalen Usability-Programm zu

[90] *vgl. Li 2003 S.4, Wild, Herges 2000 S.27 oder Kapitel 3.3.2*

beginnen, im Regelfall keinen kosten-nutzen-optimalen Einsatz von Ressourcen. Sie neigen also dazu, die Kosten einer IT-Infrastruktur überproportional zu erhöhen. Diese Feststellungen gelten unabhängig von der verwendeten Client-Art, also auch in bestehenden Systemen. Schnittpunkte zwischen den beiden Konzepten ergeben sich lediglich bei den *End User Operations* als Teil der unbudgetierten Kosten des TCO-Modells. In diesem Bereich weisen die Unterkategorien *Self-* bzw. *Peer to Peer-Support*, *Lernen im Arbeitsalltag* und die *formalen Schulungsmaßnahmen* Teilaspekte auf, die mit der Usability des eingesetzten Systems zusammenhängen. Allerdings werden hier im Rahmen einer TCO Untersuchung keine detaillierten Produktivitätsbetrachtungen vorgenommen wie in einem Usability-Programm. Während zur Usability-Betrachtung zahlreiche umfangreiche Instrumentarien zur Verfügung stehen, um Untersuchungen in diesen Bereichen durchzuführen, werden die oben erwähnten Kategorien im TCO-Modell lediglich auf Basis von Interviews erhoben. Aus diesem Grund ist davon auszugehen, dass die ermittelten Werte beim TCO-Modell deutlich unterhalb von denen der Usabilitybetrachtung liegen werden. Somit werden sich diese konzeptionellen Gemeinsamkeiten beider Betrachtungsweisen quantitativ kaum bemerkbar machen.

Die Herausforderung besteht also darin, eine geeignete Mischung zwischen beiden Konzepten zu finden. Wenn sie konsequent jeweils isoliert angewandt werden, ist bei Anwendung des TCO-Konzeptes ein günstiges, aber veraltetes, und bei einer Fokussierung auf Usability ein ideal zu bedienendes aber möglicherweise sehr teures System die Folge. In diesem Zusammenhang sind die folgenden, zwischen verschiedenen Benutzergruppen unterscheidenden Effizienzbetrachtungen ein geeignetes Mittel, um die Bedeutungen der verschiedenen Aspekte darzustellen.

3.4.2 Effizienzbetrachtungen

Die Bedeutung von Usability und TCO für die oben identifizierten Benutzergruppen wird in diesem Abschnitt untersucht. Die Usability wird zu diesem Zweck in die im Abschnitt 2.2.1 vorgestellten Teilkomponenten aufgegliedert. Bei den TCO werden zum einen die zu erwartenden Benutzerzahlen der verschiedenen Gruppen als auch die bekannte Gliederung der TCO als Ordnungskriterium verwendet, um die Verantwortlichkeiten für die entstehenden Kosten darzustellen.

3.4.2.1 Bedeutung von Usability-Komponenten für verschiedene Benutzergruppen

Idealerweise sollten alle Teilkomponenten der Usability voll erfüllt sein. Dennoch müssen in der Praxis, auch auf Grund der eingesetzten Client-Art, oft Kompromisse für beziehungsweise gegen eine bestimmte Komponente der Usability gemacht werden. Auch die unterschiedlichen Anforderungen verschiedener Benutzergruppen an die Benutzeroberfläche einer Anwendung können nur bis zu einem gewissen Grad simultan erfüllt werden[91]. Eine Analyse der Bedeutung der verschiedenen Komponenten für den zu erwartenden Benutzerkreis bietet somit eine wertvolle Grundlage für die Entscheidung solcher Kompromisse.

Die Anforderungen der verschiedenen Benutzergruppen an die Teilkomponenten der Usability unterscheiden sich, im Fall von Wissensdatenbanken, hauptsächlich zwischen den internen und den externen Benutzern. Während die *Guessability* bei externen Benutzern von entscheidender Bedeutung für die initiale Annahme des Informationsangebotes ist, spielt dieser Aspekt bei den beiden internen Benutzergruppen eine untergeordnete Rolle, da diese ohnehin regelmäßig mit der Wissensdatenbank arbeiten und so von einer guten *Guessability*, auf den gesamten Nutzungszyklus bezogen, nur sehr begrenzt profitieren können.

Ähnlich, wenn auch nicht ganz so stark ausgeprägt, verhält es sich mit den Usability-Teilkomponenten der *Learnability* und der *Re-Usability*. Erstere wird mit zunehmender Erfahrung bei der Nutzung der Wissensdatenbank auf Grund der hohen Nutzungsfrequenz interner Mitarbeiter zeitlich sehr schnell weitgehend bedeutungslos. Sie ist also nur in einem relativ kurzen Zeitraum nach der Neueinführung eines Mitarbeiters relevant. *Re-Usability* sollte bei der beschriebenen internen Nutzung überhaupt nicht in Erscheinung treten. Eine Ausnahme stellt in diesem Zusammenhang die im Abschnitt 3.1 beschriebene Untergruppe der Wissensarbeiter - die Projektmitarbeiter - dar. Auf Grund der unregelmäßigen und zeitlich begrenzten Nutzung der Wissensdatenbank durch diese Gruppe profitiert diese sowohl von einer guten *Learnability* als auch von der *Re-Usability* deutlich stärker. Beide Aspekte können für diese Benutzergruppe dazu beitragen, Einarbeitungszeiten, zum Beispiel bei der Bildung einer neuen Projektgruppe, zu verkürzen. Für externe Benutzer ist eine gute *Learnability* dann wichtig, wenn zusätzlich zu den Grundfunktionen, die "guessable" sein sollten, erweiterte Funktionalitäten

[91] *vgl. Nielsen 1993 S.41 ff.*

angeboten werden. Im Rahmen einer nur lesend genutzten Wissensdatenbank könnte dies etwa eine erweiterte Suchfunktion sein, mit deren Hilfe komplexe Suchanfragen gestellt werden können. In diesem Zusammenhang gewinnt auch die *Re-Usability* für externe Benutzer an Bedeutung, da so der einmal erlernte Umgang mit erweiterten Funktionen auch nach einem gewissen Zeitraum der Inaktivität nicht verloren geht. Dies führt dazu, dass externe Benutzer, welche die Wissensdatenbank öfter, aber azyklisch nutzen, schnell wieder zu Ergebnissen kommen, was die Akzeptanz der Wissensdatenbank erhöht.

Ein umgekehrtes Bild zeigt sich bei der Betrachtung der Bedeutung der *Experienced User Performance (EUP)* und des *System Potentials* für die verschiedenen Benutzergruppen. Für externe Benutzer sind diese beiden Teilkomponenten der Usability weniger wichtig, da erwartet wird, dass in dieser Benutzergruppe nur ein sehr geringer Anteil überhaupt zu einem erfahrenen Benutzer der Wissensdatenbank wird. Dagegen ist die *EUP* für die Benutzergruppe der Wissensarbeiter der wichtigste Usability-Faktor, da diese Gruppe zum einen das Niveau des erfahrenen Benutzers innerhalb kürzester Zeit erreicht und zum anderen die größte interne Benutzergruppe darstellt. Die Produktivität in diese Gruppe ist also von entscheidender Bedeutung. Darüber hinaus ist eine gute Usability in dieser Benutzergruppe bedeutsam für die Akzeptanz der Wissensdatenbank, die wiederum ein entscheidendes Kriterium für die inhaltliche Qualität dieser ist. Eine in den Arbeitsablauf integrierte, einfache und zeitsparende Benutzung der Wissensdatenbank ist die Vorraussetzung dafür, dass die Mitarbeiter diese intensiv und vor allem auch schreibend nutzen und so wertvolles Wissen dokumentieren. Mitarbeiter im Wissensmanagement profitieren ähnlich wie die Gruppe der Wissensarbeiter von einer hohen *EUP*, jedoch ist die Bedeutung tendenziell etwas geringer, da in dieser Gruppe das explizite Management von Wissen eine Haupttätigkeit darstellt. Somit sind die Anforderungen in Bezug auf die Integration in die sonstige Arbeitsumgebung weniger kritisch. Im Gegensatz dazu sind die Anforderungen an das *System Potential* jedoch höher als bei den Wissensarbeitern. Dies liegt in der Fülle der durch die Mitarbeiter im Wissensmanagement wahrzunehmenden Aufgaben begründet. Während die Wissensarbeiter hauptsächlich Routinetätigkeiten durchführen, beschäftigen sich Mitarbeiter im Wissensmanagement intensiver mit der Wissensdatenbank. Beispielsweise sind sie im administrativen Bereich oder in der Neugliederung von Beiträgen tätig und entwickeln damit ein tiefergehendes Verständnis der Wissensdatenbank, so dass sie in der Lage sind, ein höheres Leistungsniveau als die typische *EUP* eines Wissensarbeiters zu erreichen.

In der unten dargestellten Abbildung 3-3 sind noch einmal qualitativ die verschiedenen Bedeutungen der einzelnen Usability-Komponenten für die Benutzergruppen festgehalten. In der Grafik werden sowohl die drei identifizierten Hauptbenutzergruppen als auch die ebenfalls in Abschnitt 3.1 definierte Untergruppe der Projektmitarbeiter für die Benutzergruppe der Wissensarbeiter dargestellt.

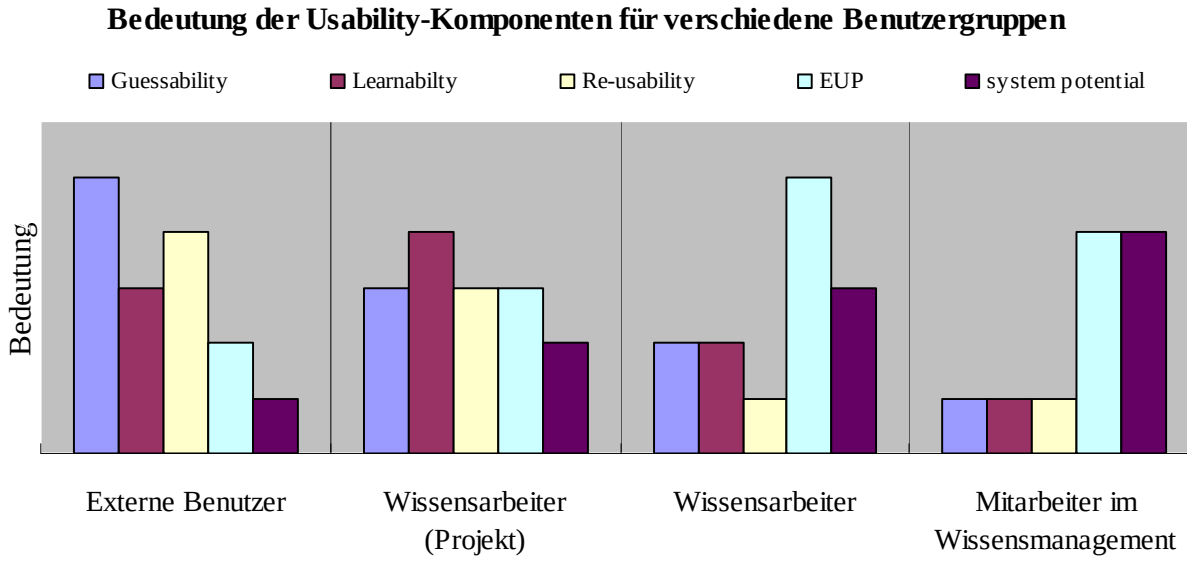


Abbildung 3-3 - Bedeutung der Usability-Komponenten für verschiedene Benutzergruppen

3.4.2.2 Wichtigkeit von TCO-Aspekten für verschiedene Benutzergruppen

Konventionelle TCO-Modelle erfassen als rein kostenorientierte Rechnungen nicht den durch eine Investition in die IT-Infrastruktur entstehenden Nutzen. Da die Kosten, die durch die Clients externer Benutzer entstehen, für eine Organisation irrelevant sind, eignet sich das TCO-Modell alleine nicht für eine Beurteilung. Reine TCO-Aspekte ergeben sich in diesem Zusammenhang maximal aus den Kosten für die Sicherung des eigenen Systems gegen unerlaubte Benutzung, oder durch die mit der Bereitstellung des "externen Informationsservices" einhergehenden Kosten, beispielsweise für genutzte Bandbreiten. Aus diesem Grund sollte die Entscheidung über die Art des Zugriffs für externe Benutzer zunächst einmal danach gefällt werden, ob mit einer bestimmten Zugangsart für externe Benutzer diejenigen Ziele erreicht werden können, die hinter einer solchen Veröffentlichung von Informationen stehen.

Die Gruppe der internen Benutzer lässt sich hingegen sehr gut anhand von TCO-Aspekten beurteilen. Obwohl geringe TCO eine Anforderung ist, die auf beide internen

Benutzergruppen gleichermaßen zutrifft, ergeben sich dennoch Unterschiede in der Bedeutung für die Wissensarbeiter gegenüber den Mitarbeitern im Wissensmanagement. Erstere stellen sowohl zahlenmäßig die größere Gruppe dar und sind zusätzlich direkt in die wertschöpfenden Prozesse einer Organisation eingebunden. Daraus lässt sich folgern, dass diese Mitarbeiter die Wissensdatenbank nur nutzen sollen und dabei von dieser möglichst wenig in ihrem normalen Arbeitsablauf unterbrochen werden sollen. Im TCO-Modell manifestiert sich diese Forderung in den indirekten Kosten, speziell in der *Downtime* und in den Kosten für den *Self-Support*. Die Ineffizienzen, die in diesen beiden Punkten anfallen können, sind in dieser Benutzergruppe besonders hoch, da Wissensarbeiter zum einen technische Probleme im Zusammenhang mit der IT-Infrastruktur meist weniger effektiv lösen können als speziell hierfür ausgebildete Mitarbeiter, und zum anderen der Arbeitsausfall bei Wissensarbeitern auf Grund der hohen Kosten der typischerweise gut ausgebildeten Mitarbeiter besonders schwer wiegt. Darüber hinaus führt die verhältnismäßig große Anzahl von Mitarbeitern in dieser Gruppe dazu, dass sowohl oben beschriebene Kosten als auch die übrigen Kostenarten des TCO-Modells sich in der Gesamtrechnung stark bemerkbar machen. Dieser Multiplikatoreffekt ist bei der Benutzergruppe der Mitarbeiter im Wissensmanagement deutlich kleiner, so dass in diesem Bereich die Kosten eines einzelnen Arbeitsplatzes für einen "Wissensmanager" in der Gesamtrechnung weniger ins Gewicht fallen. Zusätzlich sind die Benutzer in dieser Gruppe auf Grund ihres Tätigkeitsprofils intensiver mit der Wissensdatenbank vertraut. Daher ist zu erwarten, dass die Lösung technischer Probleme in dieser Gruppe deutlich effizienter vonstatten geht; ja sogar zum Tätigkeitsprofil eines Mitarbeiters im Wissensmanagement gehört.

3.5 Rich-Clients der zweiten Generation als Lösungsansatz

Die dargestellten Vorteile der beiden Client-Arten – Thin- und Thick-Clients - zu verbinden, ohne dabei die spezifischen Nachteile zu übernehmen, ist das Ziel einer neuen Client-Art. Da sich diese Client-Art noch in der Entwicklung befindet, hat sich noch kein allgemein verbindlicher Name für sie herausgebildet. Stattdessen existieren mehrere, größtenteils herstellerabhängige Bezeichnungen, die teilweise eng mit der für das Konzept angewandten Technologie zusammenhängen. So bezeichnet *Microsoft* diese neuen Clients als *Smart Clients*, *Macromedia* als *Next-Generation Rich Clients*, *IBM* als *Managed Client* und die *Gartner Group* schließlich als *High-Fidelity Client*. In dieser Arbeit wird der von *Forrester* kreierte Begriff *Rich-Clients der zweiten Generation verwendet (RC²)*, da er

sowohl technologieunabhängig ist als auch eine Weiterentwicklung in der Client-Technologie impliziert.

Im Folgenden werden im Abschnitt 3.5.1 zunächst die grundsätzlichen Ziele von *RC²* und die Bandbreite der unterschiedlichen Lösungsansätze zu Erreichung dieser Ziele vorgestellt. Abschließend wird die Anzahl der betrachteten Lösungsansätze eingeschränkt, um an Hand einer Gruppe in Abschnitt 3.5.2 die Eigenschaften und Funktionen von *RC²* herauszuarbeiten. Abschließend wird im Abschnitt 3.5.3 aufgezeigt, wie von diesen Eigenschaften und Funktionen beim Einsatz von *RC²* als Client für Wissensdatenbanken profitiert werden kann.

3.5.1 Motivation und Einordnung von Rich-Clients der zweiten Generation

RC² verfolgen grundsätzlich zwei Hauptziele: Zum einen soll dem Benutzer eine, ähnlich wie bei klassischen Thick-Clients, reiche Oberfläche zur Interaktion mit der Anwendung geboten werden. Gleichzeitig soll dieses Ziel mit einem Aufwand erreicht werden, der vergleichbar mit dem einer Web-Anwendung ist. Zur Erreichung dieser Hauptziele werden unterschiedliche technische Lösungen verfolgt, die sich in einem Kontinuum zwischen den klassischen Web-Clients und den konventionellen Thick-Clients einordnen lassen. Bedingt durch diesen technischen Hintergrund unterscheiden sich die unterschiedlichen technischen Lösungen auch in den durch den Client realisierbaren Funktionen. Folgende Abbildung 3-4 illustriert das Kontinuum der RC². Anschließend werden die verschiedenen Arten von RC² vorgestellt und entsprechende Beispiele gegeben.

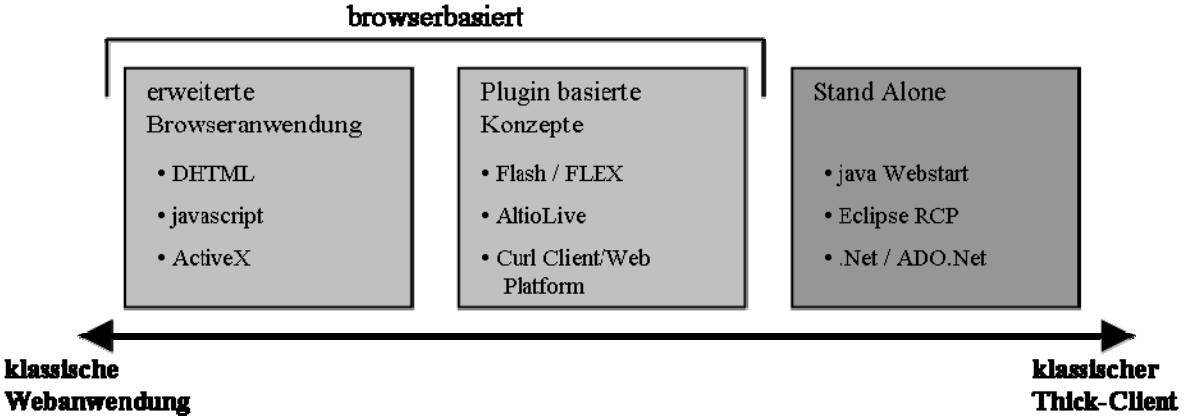


Abbildung 3-4 - Kontinuum von Rich Clients der zweiten Generation

3.5.1.1 Browser-basierte Konzepte

Bei dieser Entwicklungsrichtung handelt es sich um die Weiterentwicklung der bisherigen Web-Anwendung. Der Browser wird weiterhin als Zugangsinstrument zum Client genutzt, wodurch diese Art von Clients die kostenmäßigen Vorteile der Web-Anwendungen nutzen. Durch diese starke Bindung an den Web-Browser unterstützen browser-basierte RC² Konzepte zumeist nur reine Online-Szenarien.

Im Bereich der browser-basierten RC² lassen sich zwei Konzepte identifizieren: Zum einen die erweiterten Browser-Anwendungen, und zum anderen so genannte plugin-basierte Lösungen, die beide anschließend kurz beschrieben werden.

Erweiterte Browser-Anwendung

Diese Art der RC² ist am engsten mit der klassischen Web-Anwendung verwandt und von daher nur schwer von dieser abzugrenzen. Doch selbst wenn sie nur als nächste Evolutionsstufe der bisherigen Web-Anwendung gesehen wird, ist eine Betrachtung sinnvoll, da die in erweiterten Browser-Anwendungen realisierbaren Funktionalitäten (im Bereich der Usability) eine Mindestanforderung darstellen, die andere RC²-Konzepte erreichen müssen, um im Wettbewerb mit diesen Anwendungen bestehen zu können.

Charakteristisch für erweiterte Browser-Anwendungen sind Verbesserungen in der Benutzeroberfläche der Anwendung. Diese werden durch eine Dynamisierung der Web-Seite erreicht. So stellt in einer erweiterten Browser-Anwendung eine angezeigte Web-Seite nicht ein statisches Konstrukt dar, das lediglich dazu dient, Eingaben an den Web-Server weiterzuleiten, sondern ist ihrerseits dynamisch und enthält im begrenzten Umfang die Logik des derzeitigen Anwendungskontextes. Für den Benutzer macht sich dies vor allem dadurch bemerkbar, dass die Server-Roundtrips, und damit die Wartezeiten, erheblich reduziert werden. Da die durch Server-Roundtrips bedingten Wartezeiten von vielen Autoren als der entscheidende Schwachpunkt in der Usability von derzeitigen Web-Anwendungen gesehen werden[92], ist diese Verbesserung als die wertvollste Neuerung bei den erweiterten Browser-Anwendungen zu sehen. Ein typisches Beispiel für diese Dynamisierung der Web-Seiten ist die Verwendung von Registerkarten innerhalb des Browsers, die ohne erneuten Zugriff auf den Server gewechselt werden können. Diese

[92] *vgl. hierzu Sulzmaier 2002 S.23: "wirken sich extrem negativ auf den wahrgenommenen Grad der Usability aus" oder Spolsky 2001 S.125: „On the Web, every click results in a round-trip to the server, which reduces usability. Design to minimize round-trips“*

reagieren somit umgehend, ohne dass durch ein erneutes Laden der Seite und die dazugehörige Wartezeit die Metapher der Registerkarte zerstört wird. Selbstverständlich bleiben hierbei die auf der nun verdeckten Registerkarte vorgenommen Einstellungen und Dateneingaben erhalten (vgl. Abbildung 3-5). Weitere Reduzierungen der Wartezeiten werden auch dadurch erreicht, dass Inhalte dynamisch nachgeladen werden, während die Seite schon im Browser angezeigt wird.

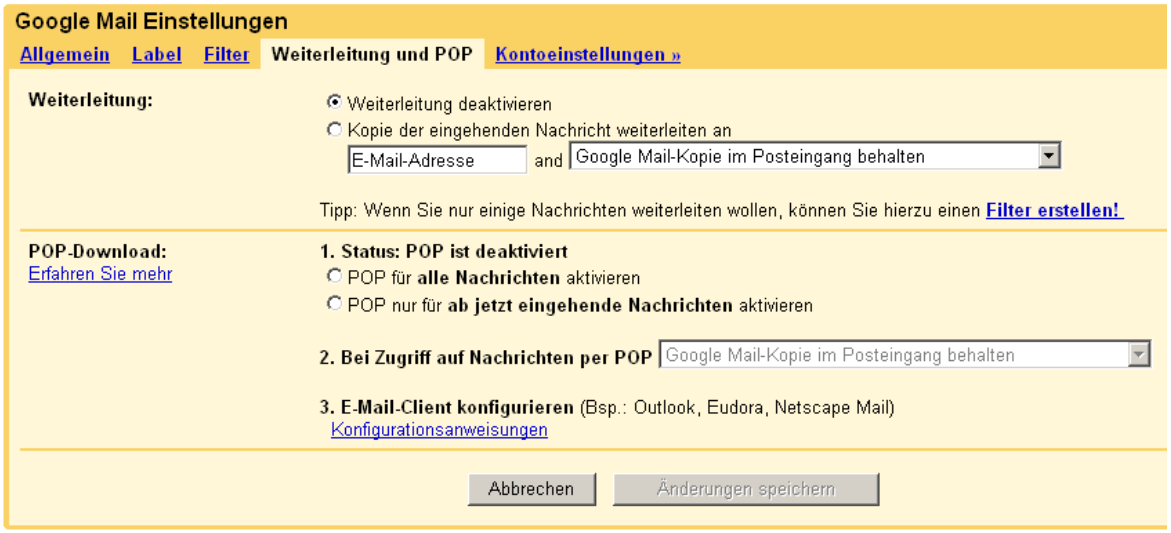


Abbildung 3-5 - Beispiel GMail - Registerkarten ohne Nachladen

Neben den verkürzten Wartezeiten gegenüber einer klassischen Web-Anwendung können erweiterte Browser-Anwendungen noch weitere Funktionen bieten, welche die Usability erhöhen. So existieren im Bereich des *Feedbacks* beispielsweise eigene Statusanzeigen, die dem Benutzer über den derzeitigen Zustand der Anwendung informieren. Auch die *User Control* kann durch erweiterte Browser-Anwendungen verbessert werden. So bieten verschiedene, erweiterte Browser-Anwendungen die Möglichkeit neben der für Web-Anwendungen üblichen Maussteuerung auch Tastaturkürzel als alternative Bedienmöglichkeit zu verwenden.

Trotz dieser Verbesserungen bleiben auch bei erweiterten Browser-Anwendungen einige Usability Probleme ungelöst. Insbesondere im Bereich der *Consistency* und der *Consideration of User Resources* lassen sich keine wesentlichen Verbesserungen erzielen, da die Anwendung immer noch durch die Grenzen des Browsers beschränkt ist, und somit eine visuelle und funktionale Integration in das umgebende Betriebssystem unmöglich

ist[93]. Ebenso bleibt das Problem der *Error Prevention* und *Recovery*, selbst wenn es durch intelligente, clientseitige Validierungsmechanismen abgemildert werden kann, zu einem großen Teil bestehen, da die Anwendung weiterhin nur eine beschränkte Kontrolle über die umgebende Ausführungsumgebung (Browser) hat.

Ein weiteres Problem der erweiterten Browser-Anwendungen besteht in der technischen Realisierung dieser Anwendungen. So erfolgt die Programmierung der erweiterten Funktionalitäten mit *JavaScript* oder auch *ActiveX*. Hierdurch entsteht zum einen ein Sicherheitsproblem: So empfiehlt das Bundesamt für Sicherheit in der Informationstechnik (BSI) Inhalte-Anbietern, auf aktive Inhalte zu verzichten[94], oder zumindest eine Version ihres Services ohne aktive Inhalte anzubieten, wodurch die erweiterten Funktionalitäten wieder nicht nutzbar sind. Zum anderen ergibt sich, bedingt durch die unterschiedlichen Browser mit denen eine solche Anwendung aufgerufen wird, die Notwendigkeit, die unterschiedlichen Anforderungen dieser einzelnen Browser zu berücksichtigen.

Dies bedeutet sowohl, dass die unterschiedlichen Dialekte und Unterstützungen, die die einzelnen Browser für die eingesetzten Technologien anbieten, bei der Programmierung einer solchen Anwendung explizit bedacht werden müssen, als auch, dass Realisierungen existieren müssen, die es ermöglichen, die Anwendung trotz fehlender Unterstützung für einige Technologien nutzbar zu halten. In der Praxis führt dies oft dazu, dass die volle Funktionalität der Anwendung nur mit einem bestimmten Browser-Produkt abgerufen werden kann, und andere Browser nur eine Teilmenge der Usability-Vorteile nutzen können. Auch können installierte Browser-Plugins dazu führen, dass einige Funktionen (z.B. Tastaturkürzel die sowohl für das Plugin als auch durch die Web-Anwendung belegt werden) der Web-Anwendung nicht funktionieren[95].

Die Programmierung einer erweiterten Browser-Anwendung ist aus den o.g. Gründen sehr komplex. Erschwerend kommt hinzu, dass für die Funktionalitäten, die durch diese Anwendungen typischerweise angeboten werden, noch kein standardisiertes Framework existiert, das die Nutzung dieser Funktionen in der eigenen Anwendung unterstützt. Dieses Fehlen eines Standards ist aus zwei Gesichtspunkten negativ zu bewerten: Zum einen erhöht es den Aufwand bei der Programmierung der Anwendung, und zum anderen existiert so keine einheitliche Vorgehensweise, wie diese erweiterten Funktionalitäten

---

[93] *vgl. Kumar 2004 für ein Beispiel wie Drop-Down-Boxen in GMail als Ersatz für Menüs verwendet werden*

[94] *vgl. BSI 2000 Maßnahme 10*

[95] *vgl. Pilgrim 2004 für eine Betrachtung der GMail Oberfläche aus der Sicht Alternativer Browser*

realisiert werden. Dies erhöht die Gefahr von negativen Wechselwirkungen mit den zahlreichen Browsern und Browser-Erweiterungen.

Beispiele für erweiterte Browser-Anwendungen sind *GMail* oder die von *Isomorphic Software* entworfenen Beispielanwendungen, die auf einem selbst entworfenen proprietären Framework basieren.

Plugin-basierte Konzepte

Plugin-basierte RC² nutzen den Web-Browser lediglich als Ablaufumgebung für ihr, auf den jeweiligen Browser zugeschnittenes, Plugin. In diesem Plugin läuft wiederum die eigentliche Anwendung. Da innerhalb des Plugins ein eigenes, definiertes Programmiermodell zur Verfügung steht, lassen sich mit Hilfe dieser Technik Benutzeroberflächen gestalten, die nur durch die Möglichkeiten des jeweiligen Plugins begrenzt sind. Dadurch können auch komplexe Operationen wie z.B. die Sortierung von Objekten im Client ohne Server-Unterstützung durchgeführt werden. In der Praxis bedeutet dies, dass die heutigen Plugin-Konzepte im Hinblick auf die Anzahl und Komplexität der zur Verfügung stehenden Widgets und den programmatischen Möglichkeiten den klassischen Programmiersprachen nahezu ebenbürtig sind. Da die Benutzeroberfläche und deren Logik nur einmalig übertragen werden und lediglich für das Nachladen von Daten die Netzwerkverbindung benötigt wird, ist es möglich, dem Benutzer eine schnell reagierende, einheitliche Oberfläche zu präsentieren, die den Anforderungen an die Usability weitgehend entspricht.

Schwächen in der Benutzeroberfläche sind lediglich im Bereich der *Consistency* festzustellen. Dieser Kritikpunkt hat seinen Ursprung in der schwierigen Integration der Plugin-Anwendung in die bestehende Betriebssystemumgebung, die im Vergleich zu den rein browser-basierten Konzepten zwar mehr Möglichkeiten bietet, aber dennoch weiterhin einen Schwachpunkt darstellt. So fügt sich die Benutzeroberfläche weder vom Design noch von den angebotenen Funktionen (z.B. Größenveränderungen von Fenstern) nahtlos in das Betriebssystem ein. Auch ist die Kommunikation mit externen Anwendungen (beispielsweise durch Drag-and-Drop) kaum realisierbar.

Ein weiterer Nachteil dieses Konzeptes ist die Abhängigkeit von dem verwendeten Plugin. Dieses muss zusätzlich zum Browser auf dem Client installiert sein und unter Umständen auch auf dem aktuellen Stand gehalten werden, um ein Arbeiten mit der Anwendung zu ermöglichen. Obwohl die Plugin-Konzepte zumindest technisch für einen begrenzten

Offline-Betrieb geeignet sind, wird diese Client-Art, vermutlich auf Grund der engen Kopplung zwischen Plugin und Browser, bisher ausschließlich in Online-Szenarien eingesetzt.

Beispiele für Technologien, die ein Erstellen dieser Client-Art ermöglichen, sind *Macromedia FLEX*, *Altio* und *Curl*. Während es sich bei *Macromedia FLEX* um eine Weiterentwicklung des ursprünglich rein für Animationszwecke entworfenen *Macromedia Flash* handelt, sind die weiteren Alternativen nur mit Blick auf diese Client-Art entwickelt worden. Die derzeit wichtigste Domäne dieser Art von Anwendungen ist die des e-Commerce. Hier ermöglichen die plugin-basierten Konzepte eine interaktive und damit ansprechendere Präsentation von Produkten (siehe z.B. Web-Seite von Mini USA). Aber auch Anwendungen für Instant-Messaging werden auf Basis dieser Technologien bereitgestellt (z.B. ein ICQ Client). An letztgenanntem Beispiel werden die Vorteile von RC² besonders deutlich, da diese Anwendung ohne vorherige Installation Funktionalitäten bietet, die bisher den reinen Thick-Client Anwendungen vorbehalten waren. All diese Beispiele nutzen auf Grund des großen Benutzerkreises und der hohen Verbreitungsquote des *Flash* Plugins ausschließlich die *Macromedia* Technologie. Die *GIGA Group* geht zwar ebenfalls davon aus, dass auf Grund der oben genannten Voraussetzungen *Macromedia* beste Chancen hat, in diesem neuen Markt ein große Rolle zu spielen, doch werden auch Chancen für die anderen Anbieter gesehen. Besonders, wenn es um firmen-interne Lösungen und damit einhergehend um geschlossene Benutzergruppen geht, werden anderen Anbietern Chancen eingeräumt, da diese Technologien teilweise die besseren Voraussetzungen bieten[96].

3.5.1.2 Stand-Alone Rich-Clients der zweiten Generation

Stand-Alone RC² stehen im Kontinuum der RC² eher auf der Seite der klassischen Thick-Clients. Während die zuvor erläuterten browser-basierten Konzepte dadurch gekennzeichnet sind, dass die Entwicklung vom Browser ausgeht und darauf aufbauend versucht wird, die Funktionalität und damit einhergehend die Usability von klassischen Thick-Client Anwendungen zu erreichen, ohne dabei die Kostenvorteile der klassischen Web-Anwendung zu verlieren, geht die Entwicklung von Stand-Alone Rich-Clients der zweiten Generation den umgekehrten Weg: Bei dieser Client-Art wird versucht, ausgehend

[96] *vgl. Meyer 2003*

von klassischen Thick-Clients, diejenigen Elemente zu verbessern, die zu den kostenmäßigen Nachteilen der klassischen Thick-Clients führen, ohne dabei die Usability des Clients zu beinträchtigen.

Aus dem dargestellten Konzept ergibt sich, dass diese, aus den klassischen Thick-Clients hervorgegangene Client-Art, in Bezug auf Usability-Eigenschaften in der Lage ist, alle Anforderungen zu erfüllen, die auch an klassische Thick-Clients gestellt werden können. Da es sich um Stand-Alone Lösungen handelt, die direkt in der Betriebssystemumgebung ausgeführt werden, entfallen die Einschränkungen, die im Vergleich zu den anderen Konzepten durch den Browser als Laufzeitumgebung hervorgerufen werden. Dieser direkte Zugriff auf Ressourcen des Betriebssystems und die Unabhängigkeit vom Browser erleichtert auch die Implementierung von Offline-Fähigkeiten, da die Anwendung ohne Browser benutzt werden kann. Auch besteht so die Möglichkeit, größere Datenmengen entweder durch direkten Zugriff auf das Dateisystem oder durch eine in die Anwendung eingebettete Datenbank zu speichern.

Interessanter als die weitgehend dem Funktionsumfang klassischer Thick-Clients entsprechenden Möglichkeiten, sind bei den Stand-Alone RC² die Konzepte, welche die Kosten dieser Client-Art auf ein mit Web-Anwendungen vergleichbares Niveau reduzieren (sollen). In diesem Bereich adressieren aktuelle Lösungen insbesondere die Installation und die Wartung der Client-Anwendung, da in diesen Bereichen klassische Thick-Client Anwendungen die größten Kostennachteile gegenüber Web-Anwendungen aufweisen.

Konkret bedeutet dies, dass Stand-Alone RC² über Mechanismen verfügen, die eine einfache initiale Installation durch den Endanwender ermöglichen. Diese erste Auslieferung an den Client erfolgt zumeist mit Hilfe einer Web-Seite, über die der Client, im Idealfall innerhalb eines geführten Dialogs, installiert werden kann. Die Wartung, also insbesondere die Aktualisierung des Clients wird meist mit Hilfe eines Update-Servers und einer entsprechenden Funktion im Client automatisch vorgenommen. Hierdurch kann eine Aktualisierung des Clients auf eine neue Version mit ähnlich geringem Aufwand wie bei einer Web-Anwendung einfach durch die Bereitstellung einer neuen Version auf dem Updateserver zentralisiert vorgenommen werden. Die meisten Konzepte verfügen darüber hinaus über eine Struktur, die es ermöglicht, bei Updates nur gezielt relevante Komponenten zu aktualisieren. Dadurch kann die Netzwerkbelastung und der Zeitbedarf für solche Updates reduziert werden.

Weiterhin zeichnen sich Stand-Alone RC² dadurch aus, dass sie im Gegensatz zu ihren Vorgängern, den klassischen Thick-Clients, weniger von bestimmten Ressourcen des Zielsystems abhängig sind. Dies zeigt sich beispielsweise bei der Nutzung von Bibliotheken: Während klassische Anwendungen durch Mehrfachnutzung von Bibliotheken eine indirekte Abhängigkeit zueinander aufbauen können, beinhalten Stand-Alone RC² ihre Bibliotheken direkt, wodurch diese Abhängigkeiten vermieden werden. Hierdurch wird die Unabhängigkeit des Clients gegenüber Veränderungen des umgebenen Systems erhöht und die Ausfallzeit damit reduziert.

Die Bereitstellung der oben vorgestellten Funktionalitäten erfordert zusätzlich zum Betriebssystem eine Laufzeitumgebung, die auf den potentiellen Clients vorhanden sein muss. Diese Laufzeitumgebung ist abhängig von der eingesetzten Technologie und im Allgemeinen umfangreicher als bei den plugin-basierten Konzepten. Somit eignen sich Stand-Alone Clients weniger für Bereiche in denen viele unterschiedliche Clients bedient werden müssen (z.B. B2C Lösungen), da im Moment nicht davon ausgegangen werden kann, dass entsprechende Laufzeitumgebungen eine hinreichende Installationsbasis haben. Sie eignen sich wohl aber für Bereiche, in denen die Art und Ausstattung der Clients kontrolliert werden kann, also insbesondere Unternehmen oder ähnliche Organisationen.

Technologien mit denen Stand-Alone RC² entwickelt werden können sind zum einen das *.NET-Framework* von *Microsoft* und auf der anderen Seite *Java*-basierte Lösungen wie *Webstart* oder *das Eclipse RCP Framework*.

3.5.2 Eigenschaften von Stand-Alone Rich-Clients der zweiten Generation

Aus den oben beschriebenen Realisierungsmöglichkeiten für RC² wird deutlich, dass eine große Bandbreite technischer Möglichkeiten mit entsprechend unterschiedlichen Eigenschaften existiert. Hierbei bieten die Stand-Alone Clients auf Grund ihres Ursprungs potentiell die umfangreichsten Funktionalitäten. Aus diesem Grund fokussiert diese Arbeit im weiteren Verlauf auf Stand-Alone Clients, deren grundlegende Eigenschaften, unabhängig von der jeweiligen technischen Umsetzung, in diesem Abschnitt unter vier Oberpunkten zusammengefasst werden. Die hier angesprochenen Eigenschaften stellen eine minimale Anforderung an Stand-Alone RC² dar und sind in konkreten Produkten meist noch durch zusätzlich Funktionen erweitert.

Zentrale Administration

Zur Administration der Anwendung gehören zunächst die Installation und das Update der Clients. Ein weiterer Aspekt ist die Rechteverwaltung für Daten und Anwendungen. So muss beispielsweise sichergestellt werden, dass die jeweilige Anwendung und damit verbundenen Daten nur durch berechtigte Benutzer verwendet werden können. Diese Aufgaben müssen bei Stand-Alone RC² zentral auf dem Server, also ohne manuellen Eingriff auf den Clients, durchgeführt werden können.

Usability

Die Benutzeroberfläche muss so gestaltet sein, dass sie die jeweilige Aufgabe optimal unterstützt. Konkret bedeutet das, dass Stand-Alone RC² eine, im Vergleich zu Web-Anwendungen aus nativen Widgets bestehende, reichere Benutzeroberfläche bieten sollten. Darüber hinaus verbessert sich die Usability der Oberfläche auch dadurch, dass umfangreichere Operationen auf dem Client unterstützt werden und somit weniger Kommunikationsschritte mit dem Server erforderlich sind. Schließlich ermöglicht diese Art der Benutzeroberfläche die nahtlose Integration mit schon installierten Client-Anwendungen, da beispielsweise Drag and Drop Funktionen unterstützt werden können.

Unterstützung von Online- und Offline-Szenarien

Stand-Alone RC² ermöglichen es, mit Hilfe ein und desselben Clients eine Anwendung sowohl im Offline als auch im Online Modus zu benutzen. Mit dieser Funktionalität geht einher, dass eine lokale Datenhaltung und ein entsprechender Synchronisationsmechanismus zwischen lokalen (Offline-)Daten und globalen (Online-) Daten erforderlich wird. Zusätzlich muss der lokale Datenspeicher vor nicht autorisiertem Zugriff geschützt werden, um den Missbrauch der verteilten Daten (beispielsweise bei Verlust eines Notebooks) zu vereiteln.

Nutzung lokaler Ressourcen

Die Nutzung lokaler Ressourcen lässt sich zum einen als reine Unterstützungsfunktion für die o.g. Funktionalitäten betrachten. So kann die Anforderung der Offline-Fähigkeit bzw. die Unterstützung umfangreicherer Funktionen (z.B. Sortierfunktionen) auf dem Client nur unter Zuhilfenahme lokaler Ressourcen erfüllt werden.

Auf der anderen Seite hilft die Ausführung von Business-Logik auf dem Client, sowohl Server- als auch Netzwerkressourcen einzusparen, da zum einen seltener mit dem Server kommuniziert werden muss, und zum anderen die durch den Client übermittelten Daten

beispielsweise durch vorherige, clientseitige Validierung oder Berechnung von höherer Qualität sind und somit weniger oft zurückgewiesen werden.

3.5.3 Stand-Alone Rich-Clients der zweiten Generation als Benutzeroberfläche für Wissensdatenbanken

Wissensmanagement wird übereinstimmend als weitaus mehr als nur Technologie angesehen[97]. Dennoch spielt die Technologie, als so genannter *Enabling Factor*, beim Wissensmanagement eine wichtige Rolle und wirkt "geradezu als Katalysator der Wissensmanagement-Bewegung"[98]. Somit stellt die Technologie zwar eine notwendige, aber keine hinreichende Bedingung für ein erfolgreiches Knowledge Management (KM) dar. In diesem Zusammenhang ist das Konzept der Wissensdatenbank "einer der bekanntesten Ansätze zur Technologieanwendung im Wissensmanagement“[99]. Eine Wissensdatenbank erleichtert die Ablage, die Verteilung und das Auffinden relevanter Informationen und fördert darüber hinaus durch die personelle Zuordnung von Beiträgen zu ihren Autoren die Bildung von Netzwerken innerhalb einer Organisation[100].

Neben den im vorherigen Abschnitt beschriebenen grundsätzlichen Eigenschaften bieten RC² Vorteile, die sie gerade im Anwendungsgebiet der Wissensdatenbanken zu einer sinnvollen Client-Art machen. Da Wissensdatenbanken zu den Wissenstechnologien gehören, die von der aktiven Teilnahme eines großen Benutzerkreises abhängig sind, können die Vorteile der einfachen Installation und Wartung gut umgesetzt werden[101]. Ein weiterer Punkt betrifft die Wichtigkeit der aktiven Teilnahme der Benutzer. Hier schafft ein Stand-Alone RC² durch die mögliche Integration in das Betriebssystem (z.B. Drag and Drop oder SSO) die Vorraussetzung dafür, dass die Nutzung der Wissensdatenbank einfach, also insbesondere ohne langwierige Unterbrechung des eigentlichen Arbeitsprozesses, stattfinden kann. Diese reibungslose Integration in die Arbeitsprozesse ist für die Akzeptanz einer KM-Lösung eine der entscheidenden Komponenten, da mangelnde Akzeptanz den Erfolg ganzer KM-Projekte gefährden kann[102].

---

[97] *vgl. Davenport, Prusak 1998b S.123*

[98] *vgl. Davenport, Prusak 1998a S.241*

[99] *vgl. Davenport, Prusak 1998a S.252*

[100] *vgl. Alvesson 2004 S.174 oder McDermott 1999 S.104*

[101] *vgl. zu Problemen mit klassichen Rich Clients z.B. Schliwka 1998 S.307*

[102] *vgl. Alvesson 2004 S.179*

Darüber hinaus erfordert die Anwendung von Wissenstechnologie "in aller Regel ein interaktives und iteratives Eingreifen seitens der Benutzer"[103]. RC² bieten dem Benutzer potentiell mehr Interaktionkanäle an als klassische Web-Anwendungen und ermöglichen so einen höherwertigen Wissenstransfer[104]. Beispiele für diese Eigenschaft sind die Integration von multimedialen Inhalten oder Interaktionsmöglichkeiten mit deren Hilfe beispielsweise komplexe Navigationsstrukturen auf den in der Wissensdatenbank enthaltenen Objekten realisiert werden können.

Schließlich ermöglicht die Offline-Fähigkeit des Clients sowohl den Zugriff auf in der Wissensdatenbank gespeicherte Informationen als auch das Anlegen neuer Inhalte unabhängig von der Existenz einer Online-Verbindung. Somit können neue Informationen direkt im System erfasst werden. Die ständige Verfügbarkeit des Systems hilft, ebenfalls die Akzeptanz und die Qualität der in der Wissensdatenbank enthaltenen Informationen zu erhöhen, da generiertes Wissens zeitnah dokumentiert werden kann und dieses somit nicht durch den Zeitraum, der bis zur Verfügbarkeit einer Online Verbindung vergeht, degeneriert oder gar ganz verloren geht. Ein weiterer Vorteil der Offline-Fähigkeit ist in einem geringeren Lernaufwand bzw. einer geringeren Fehleranfälligkeit auf Seiten der Benutzer zu sehen, da für beide Betriebsarten derselbe Client benutzt werden kann.

3.6 Diskussion: Welcher Client für welche Benutzergruppe

In den vorherigen Abschnitten wurden die unterschiedlichen Eigenschaften verschiedener Client-Arten in Bezug auf Usability und TCO, sowie auf die im Bereich von Wissensdatenbanken zu erwartenden Benutzergruppen analysiert. Darüber hinaus wurden RC² als eine neue, evolutionäre Client-Art vorgestellt. In diesem Abschnitt werden die gewonnen Erkenntnisse zusammengefasst, um die Frage beantworten zu können, für welche Benutzergruppe im beschriebenen Szenario der Wissensdatenbank jeweils welche Client-Art am sinnvollsten eingesetzt werden kann.

Nachfolgende Grafik (vgl. Abbildung 3-6) visualisiert die Zuordnung von Client-Arten zu Benutzergruppen und zeigt ebenso Einflussfaktoren für diese Entscheidung.

---

[103] *vgl. Davenport, Prusak 1998a S.251*
[104] *vgl. Hartlieb 2002 S.97*

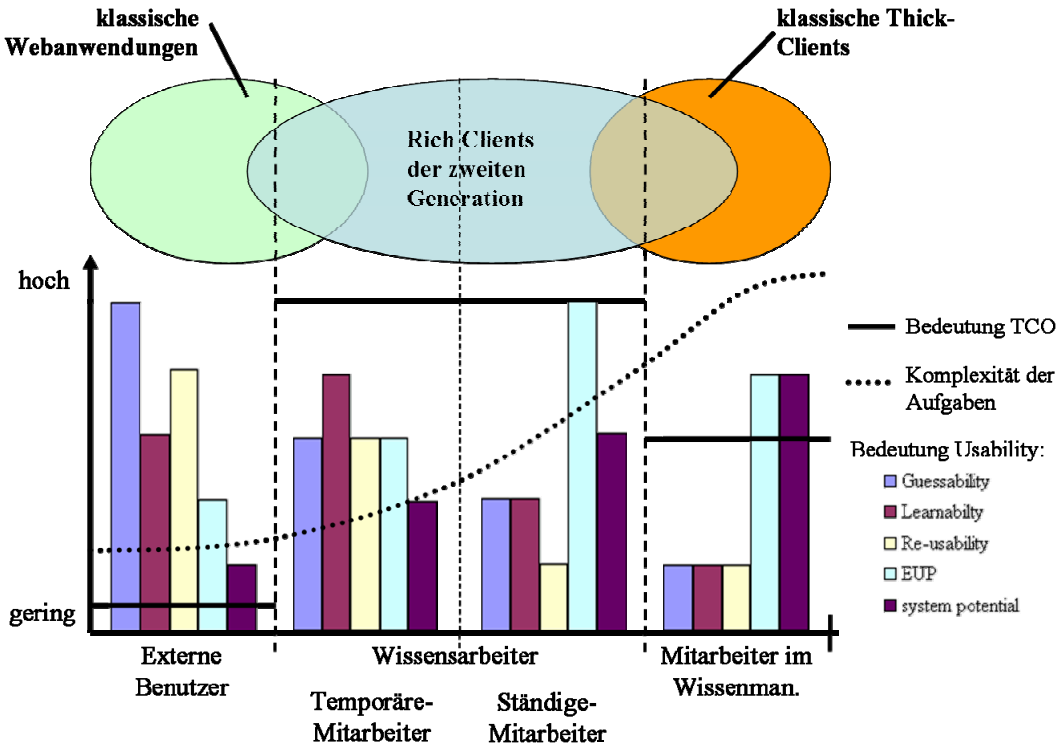


Abbildung 3-6 - Zuordnung von Client-Arten zu Benutzergruppen und Einflußfaktoren

Im Bereich der externen Benutzer ist größtenteils die klassische Web-Oberfläche, also der Browser, als Client-Art angesiedelt. Da TCO-Überlegungen in dieser Gruppe keine Rolle spielen, ergibt sich dies durch die Ziele eines solchen Informationsangebotes und den damit einhergehenden Usability-Überlegungen. Die in dieser Gruppe typischerweise sehr geringe Nutzungshäufigkeit und die Heterogenität innerhalb der Gruppe führt dazu, dass für den Zugriff ein kompatibles und ohne Vorkenntnisse zu bedienendes Verfahren benötigt wird. Zusätzlich steht externen Benutzern zumeist nur ein eingeschränkter Zugriff (z.B. ein nur lesender Zugriff) auf das System zur Verfügung, was die Komplexität der zu bearbeitenden Aufgaben reduziert. Unter diesen Vorraussetzungen stellt sich die klassische Web-Anwendung als die bevorzugte Alternative dar, während RC² oder auch die klassischen Thick-Clients, entweder eine für den einmaligen Einsatz wenig lohnenswerte (automatische) Installation erfordern, oder im Falle der erweiterten Browser-Anwendungen derzeit noch häufig zu Inkompatibilitäten mit verschiedenen Browsern führen. Beides kann die Reichweite des Informationsangebotes einschränken. Weiterhin verfügt der Browser

auf Grund seiner eingeschränkten Benutzeroberfläche[105] über Usability-Vorteile in dem für diese Benutzergruppe wichtigem Bereich der *Guessability*.

Die RC² eignen sich insbesondere für die Benutzergruppe der Wissensarbeiter. Im Vergleich zur klassischen Web-Anwendung zeichnen sich RC² vor allem durch die bessere Usability in den Teilbereichen aus, die für diese Benutzergruppe von Bedeutung sind. Da die Komplexität der durch die Wissensarbeiter wahrzunehmenden Aufgaben bei der Arbeit mit der Wissensdatenbank größer ist, benötigen diese auch eine entsprechend komplexere Benutzeroberfläche[106], die von klassischen Web-Anwendungen nicht geboten wird. Die reichere Oberfläche dieser Client-Art ermöglicht es somit den Wissensarbeitern, produktiver zu arbeiten, insbesondere auch in Bezug auf die Integration der Wissensdatenbank in die bestehende Arbeitsumgebung. Ein weiteres Argument für den Einsatz von RC² durch Wissensarbeiter ist die hohe Bedeutung der TCO in dieser Benutzergruppe, da es sich um eine große, interne Gruppe handelt. Hierdurch vervielfacht sich der TCO-Vorteil der RC² gegenüber den klassischen Thick-Clients und macht deshalb den Einsatz von RC² besonders interessant.

Der Einsatz klassischer Thick-Clients beschränkt sich auf den Einsatz bei den Mitarbeitern im Wissensmanagement. Dieser Einsatzbereich ergibt sich weniger aus einer funktionalen Notwendigkeit des Einsatzes von klassischen Thick-Clients, sondern viel mehr aus einer Kosten/Nutzen Betrachtung eines Wechsels von einem klassischen Thick-Client zu einem RC². Es wird also davon ausgegangen, dass die Mitarbeiter im Wissensmanagement bereits über einen klassischen Thick-Client zur Bearbeitung ihrer Aufgaben verfügen. Auf Grund der relativ geringen Anzahl von Benutzern die dieser Kategorie zuzuordnen sind, werden die TCO Vorteile, die mit einer Einführung von RC² einhergehen, weniger deutlich. Gleichzeitig sind die Kosten für die Implementierung eines RC² höher, als wenn dieser nur für die Wissensarbeiter entwickelt wird, da der gesamte Funktionsumfang des möglicherweise über einen langen Zeitraum gewachsenen, klassischen Thick-Clients im neuen Client abgebildet werden muss. Somit muss eine Einführung von RC² im Einzelfall geprüft werden und sollte gegenüber der Benutzergruppe der Wissensarbeiter eine deutlich geringere Priorität aufweisen. Ein anderes Bild ergibt sich, wenn eine Wissensdatenbank neu eingeführt wird. In diesem Fall ist auch für die Mitarbeiter im Wissensmanagement die

[105] *vgl. Spolsky 2001 S.129*
[106] *vgl. Diezmann 2002 S.115*

Verwendung von RC² zu empfehlen, da diese im Bereich der TCO Vorteile bieten und aus Usability-Gesichtspunkten den klassischen Thick-Clients ebenbürtig sind.

Die jeweiligen Überschneidungen zwischen den einzelnen Client-Arten verdeutlichen, dass es innerhalb der identifizierten Gruppen einzelne Benutzer oder auch kleinere Gruppen geben kann, die Anforderungen haben, welche die Wahl einer anderen Client-Art sinnvoll werden lässt. So ist beispielsweise im Bereich der externen Benutzer, die die Wissensdatenbank regelmäßig zur Information nutzen, eine alternative Zugangsmöglichkeit zu den präsentierten Informationen, auch über einen RC², als Service denkbar. Umgekehrt können insbesondere externe Projektmitarbeiter ein den externen Benutzern ähnliches Anforderungsprofil aufweisen, so dass es in solchen Fällen sinnvoll sein kann, diesem Personenkreis ebenfalls über eine klassische Web-Anwendung Zugang zur Wissensdatenbank zu verschaffen. Der Überschneidungsbereich zwischen klassischen Thick-Clients und RC² ergibt sich aus den bereits oben angesprochenen Kosten/Nutzen Überlegungen. Während bei Neuentwicklungen zunehmend auf RC² gesetzt werden sollte, ist der Überschneidungsbereich der klassischen Thick-Clients als Beibehaltung von Altsystemen auf Grund einer Kosten/Nutzen Betrachtung zu sehen

Abschließend ist festzuhalten, dass die hier dargestellten Vorschläge zur Gestaltung der Client-Struktur nur eine grobe Einordnung darstellen, und insbesondere die Überschneidungsbereiche einer Einzelfallprüfung aufgrund der Struktur und den Anforderungen der jeweiligen Organisation bedürfen. Kriterien die zur Entscheidung dieser Konflikte herangezogen werden sollten sind, neben der schon mehrfach angesprochenen möglichen Präsenz eines Altsystems, das Vorhandensein der Benutzergruppen (z.B. entfallen externe Benutzer wenn das System nur intern Anwendung findet), sowie die Anzahl der Benutzer in den jeweiligen Benutzergruppen.

4 Das Eclipse RCP Framework und der IBM Workplace

Im Rahmen dieser Arbeit wird zur Implementierung einer wissensintensiven Anwendung auf dem *IBM Workplace Managed Client* aufgesetzt. Da dieser technisch auf dem *Eclipse RCP Framework* basiert und zum Produktportofolio *IBM Workplace* gehört, sollen beide Aspekte zur besseren Einordnung anschließend kurz beleuchtet werden. Zunächst erfolgt die Vorstellung des *RCP*-Frameworks und anschließend die Erläuterung des *IBM Workplace* Konzeptes, wobei insbesondere auf den *Managed Client* eingegangen wird.

4.1 Das Eclipse RCP Framework

Das *Eclipse RCP Framework* ist mit der Version 3.0 des *Eclipse*-Frameworks im Juni 2004 eingeführt worden. Bis zu diesem Zeitpunkt stellte *Eclipse* eine zwar sehr vielseitig einsetzbare Umgebung dar, die aber dennoch hauptsächlich für die Softwareentwicklung eingesetzt wurde. Durch die flexible Plugin-Strukur ist es möglich, die Plattform mit nahezu beliebigen Tools zu erweitern. So basieren auch kommerzielle Entwicklungsumgebungen auf *Eclipse* indem sie es mit eigenen Plugins erweitern. Um die Ladezeiten beim Einbinden von mehreren hundert Plugins zu verkürzen, bestehen Plugins grundsätzlich aus zwei Teilen, einem rein deskriptiven Teil, der aus einer XML-Datei besteht, und dem eigentlich zum Plugin gehörende Code (vgl. Abbildung 4-1).

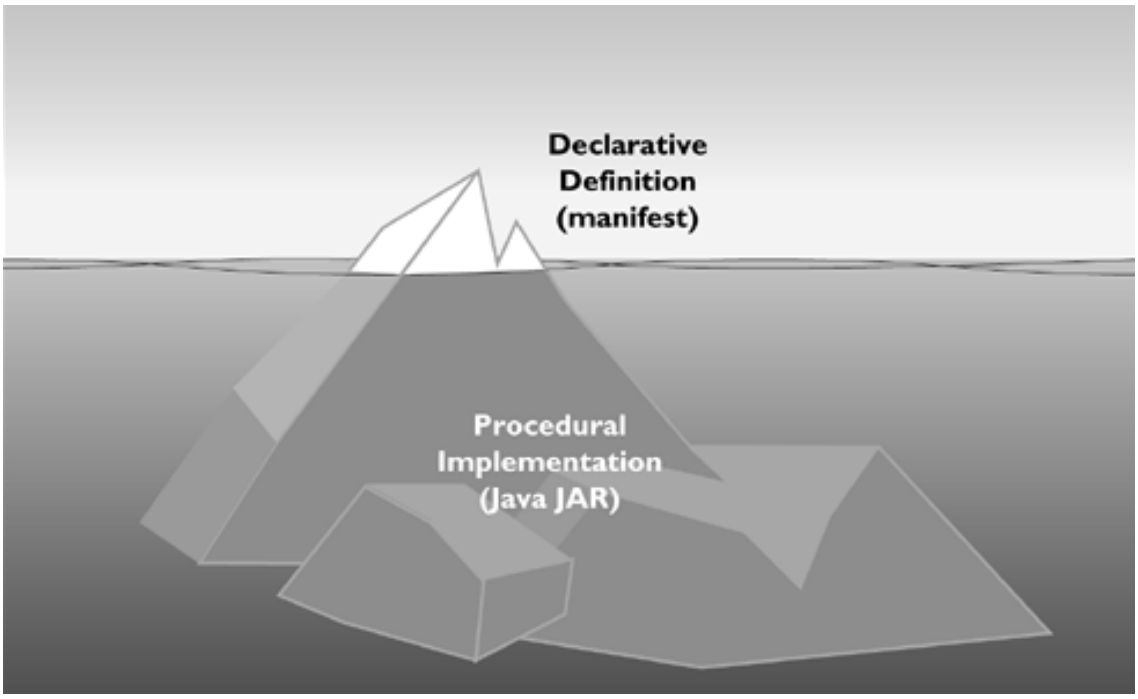


Abbildung 4-1 - Zweiteilung der Plugin-Struktur in Eclipse[107]

Hierdurch ist es möglich, die durch das Plugin bereitgestellten Funktionalitäten durch Auswertung des deskriptiven Teils in der Oberfläche zunächst anzulegen, aber den eigentlichen Code für diese Erweiterung erst zu laden und auszuführen, wenn diese Funktionalität tatsächlich aktiviert wird (*Lazy Loading*). Trotz dieser Flexibilität durch die Plugin-Struktur waren in Versionen vor 3.0 einige für eine Entwicklungsumgebung typische Funktionalitäten so tief in der Plattform verankert, dass diese "fehlende Trennung der IDE- von den allgemeinen Funktionen"[108] dazu führte, dass die Plattform nur eingeschränkt oder nur mit hohem Aufwand für beliebige Anwendungen genutzt werden konnte (vgl. Abbildung 4-2).

[107] *vgl. Gamma, Beck 2003 S.25*
[108] *vgl. Gerhardt, Wege 2004 S.44*

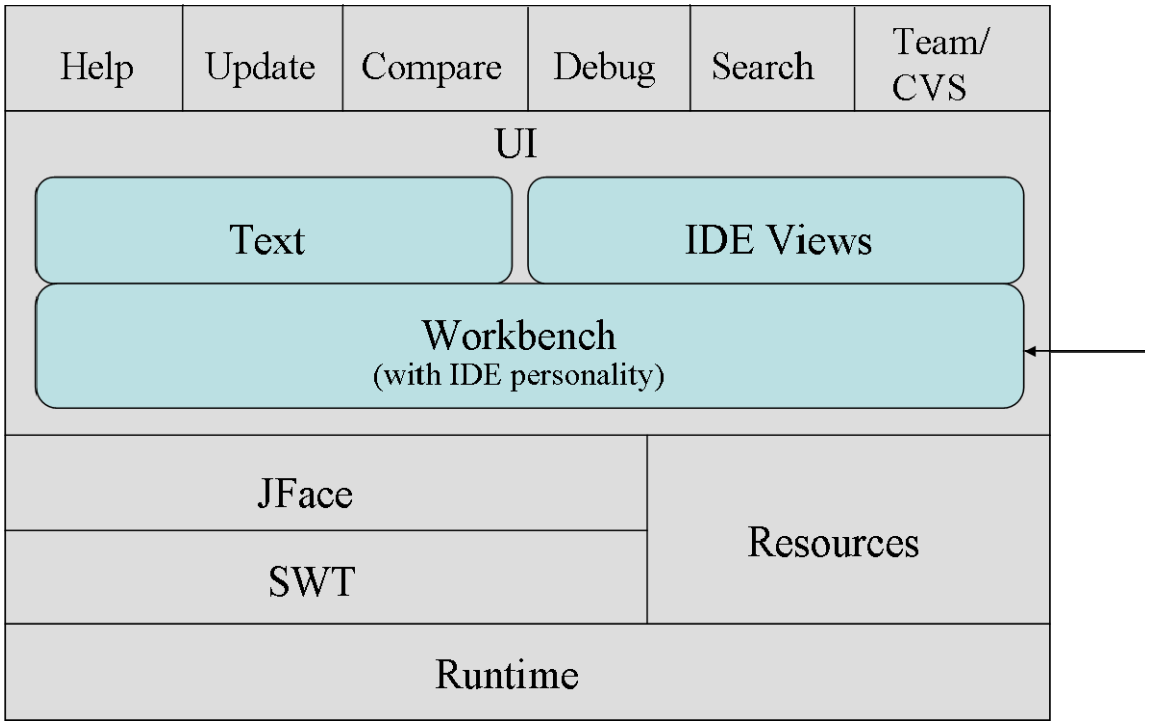


Abbildung 4-2 - Aufbau Eclipse vor der Version 3.0[109]

Mit der Version 3.0 wurden diese Abhängigkeiten beseitigt und spezielle Eigenschaften der Plattform, die bis dahin fest kodiert waren (z.B. das Vorhandensein einer Statusleiste), konfigurierbar gestaltet, und so eine Plattform geschaffen, welche aus einer minimalen Anzahl von Kern-Plugins (dem *RCP-Framework*) besteht, die beliebig erweitert werden können. Zusätzlich entspricht die Runtime mit der Version 3.0 dem *OSGi (Open Services Gateway Initiative*). Die *Eclipse* Entwicklungsumgebung ist in diesem Sinne jetzt eine spezielle Ausprägung einer *RCP*-Anwendung, da sie ebenfalls auf den Kern Plugins basiert. Die nachfolgenden Abbildung 4-3 illustriert die oben beschriebene Neuausrichtung der *Eclipse*-Plattform[110].

[109] *vgl. Edgar 2004 S.10*

[110] *für Informationen zu den einzelnen Komponenten sei auf das Eclipse Projekt verwiesen*

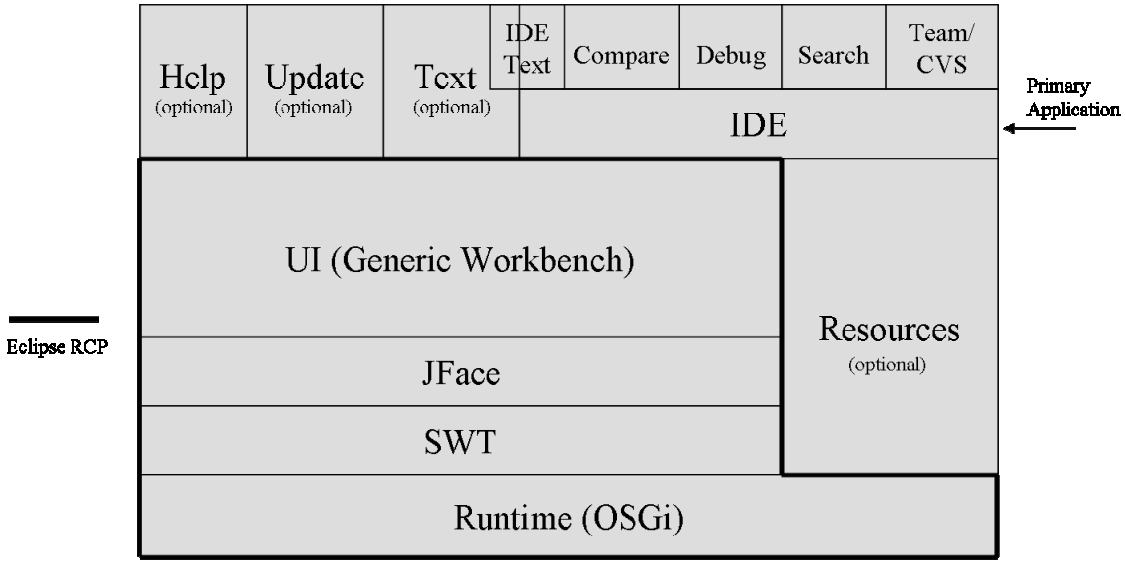


Abbildung 4-3 - Aufbau Eclipse seit der Version 3.0[111]

Das *RCP*-Framework hat eine minimale Größe von ca. fünf Megabyte, typischerweise werden jedoch auch das Framework für Updates und das Hilfeframework zur *RCP*-Plattform hinzugefügt. Zusätzlich muss, da *Eclipse* auf *Java* basiert, eine *Java Runtime Environment (JRE)* auf den jeweiligen Clients vorhanden sein.

4.2 IBM Workplace und der Workplace Managed Client

IBM Workplace bezeichnet als Oberbegriff eine neue erweiterbare Plattform in der die verschiedensten kollaborativen Anwendungen (z.B. E-Mail-, Instant Messaging-, Kalender-, Diskussions- und Dokumenten-Management-Anwendungen) ausgeführt werden. Der zentrale Gedanke beim *Workplace*-Konzept ist es, alle für eine bestimmte Aufgabe notwendigen Anwendungen in einer zentralen Oberfläche integriert zur Verfügung zu haben. Dieses Ziel der integrierten Verfügbarkeit aller relevanten Anwendungen bezieht sich auch auf die zu verwendenden Client-Alternativen. Die *Workplace*-Konzeption ermöglicht es, auf die gleichen Daten mit verschiedenen Clients zuzugreifen, wie nachfolgende Abbildung 4-4 verdeutlicht.

[111] *vgl. Edgar 2004 S.11*

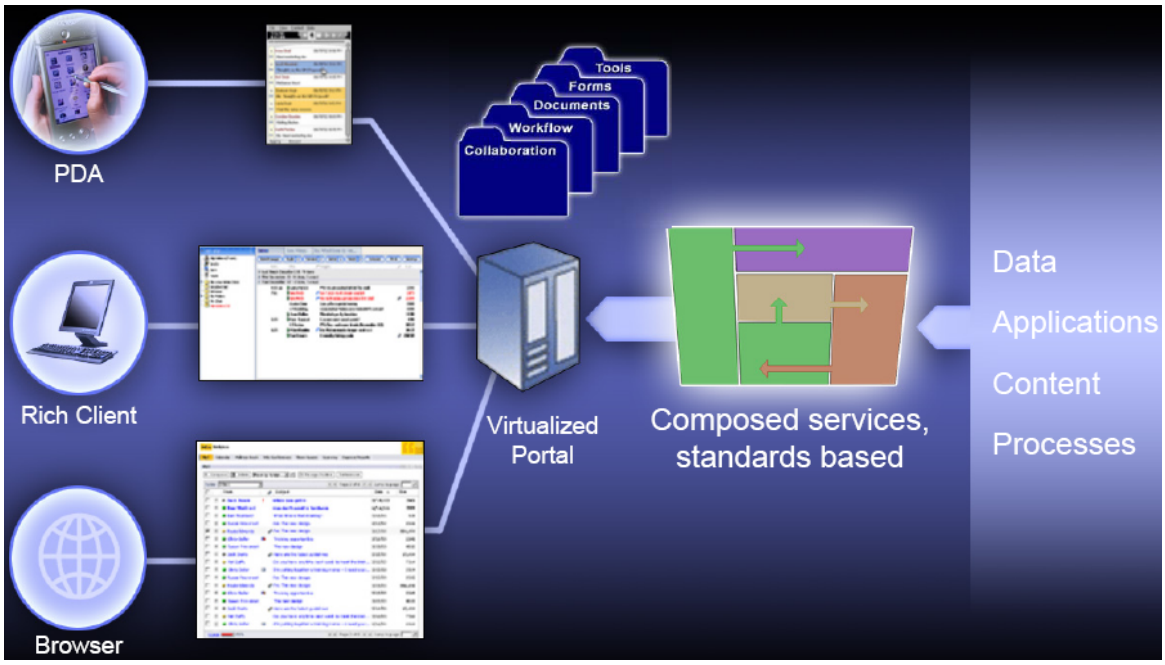


Abbildung 4-4 - Übersicht Workplace-*Konzeption*[112]

Die Rich-Client-Variante für den Zugriff auf *Workplace*-Daten, der *Workplace Managed Client (WMC)*, basiert wie oben schon angedeutet auf dem *Eclipse Rich-Client-Framework (RCP)*, ist also eine eigene *RCP*-Anwendung. Diese Anwendung unterscheidet sich vom reinen *RCP*-Framework in zwei Hauptpunkten. Zum einen werden beim *Workplace*-Client schon Anwendungen mitgeliefert. So sind die kollaborativen Anwendungen wie E-Mail, Instant Messaging, Dokumentenmanagement und weitere Funktionen schon in den Client integriert, bzw. werden beim ersten Start vom *Workplace*-Server heruntergeladen. Aber auch im Bereich der funktionalen Erweiterung bringt der WMC einiges mehr mit als die pure *RCP*-Plattform. An erster Stelle wären hier die Erweiterung im Bereich des Updatemanagements und der Benutzerverwaltung, sowie die in den Client integrierte Datenhaltung zu nennen. Zusätzlich besteht die Möglichkeit, die von den schon vorhandenen Anwendungen mitgebrachten Funktionalitäten (z.B. E-Mail), über eine API[113] programmatisch für eigene Erweiterungen des *WMC* zu nutzen. Abbildung 4-5 verdeutlicht den Aufbau des *WMC*, und macht klar, wie dieser auf der *RCP*-Plattform aufsetzt, und welche zusätzlichen Services angeboten werden.

---

[112] *vgl. Bisconti, McKinney 2005 S.33*

[113] *API: Application Programming Interface*

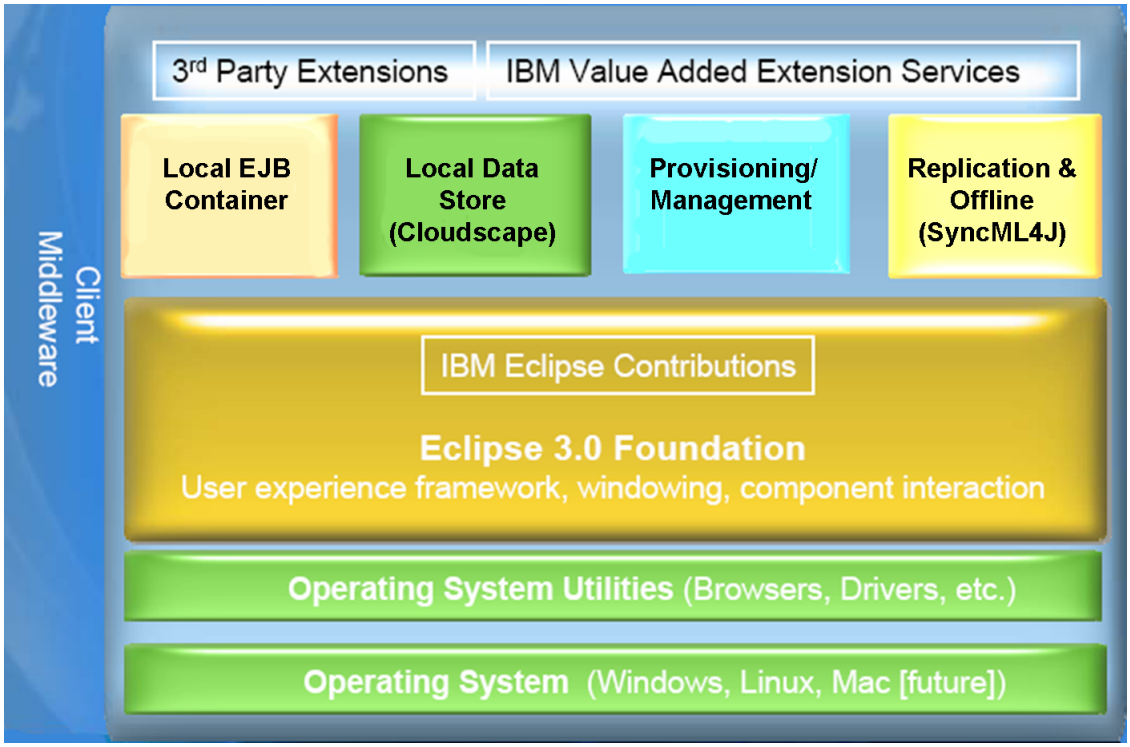


Abbildung 4-5 - Aufbau des Workplace Managed Client[114]

Trotz dieser angesprochenen Erweiterungen sind die für die *RCP*-Plattform entwickelte Plugins, auch wenn sie nicht die vollen Möglichkeiten des *WMC* ausnutzen, im *WMC* lauffähig. U. a. um diesem Umstand der unterschiedlichen Integrationsstufen Rechnung zu tragen, wurden von *IBM* verschiedene Rollen für ISV (Indepent Software Vendors) unterschieden, die es ermöglichen, externe Erweiterungen des *WMC* einzuordnen. Dieses Modell unterscheidet die in nachfolgender Tabelle 4-1 dargestellten vier Rollen[115]:

Name der Rolle	Beschreibung
Coexist	Reines Nebeneinander von Anwendungen, Integration nur aufgrund von Betriebssystemfunktionen;
Accommodate	Rein visuelle Integration, die Anwendung wird innerhalb des *WMC* angezeigt;
Leverage	Nutzung von Teilen der *WMC* Infrastruktur (z.B. Nutzung der Updatefunktion, aber eigene Datenhaltung);
Exploit	Volle Integration d.h. alle Möglichkeiten des *WMC* werden benutzt;

Tabelle 4-1 - Rollenübersicht für Independent Software Vendors (ISV)[116]

[114] *vgl. Heidloff, Rao 2005 S.7*

[115] *für eine detaillierte Darstellung sei auf Kraenzel 2004 verwiesen*

[116] *vgl. Kraenzel 2004 S. 4ff.*

5 Vorstellung der implementierten Lösung

In diesem Kapitel wird die Implementierung der Lösung beschrieben. Die vorliegende Lösung baut dabei auf zwei schon vorhandenen Systemen auf. Zum einen ist es eine Migration von Teilen der bestehenden Anwendung *Knowledge-Pool (K-Pool)*, die auf Basis eines Groupware-Systems realisiert wurde, auf eine neue Client-Technologie. Zum anderen wird für das notwendige relationale Datenmodell auf eine schon vorhandene Migration für oben genannte Anwendung, dem *K-Pool Everyplace*, zurückgegriffen.

Dementsprechend wird im Folgenden zunächst auf den Einsatzbereich und die Funktionalität des vorhandenen Groupware basierten *K-Pools* eingegangen. Anschließend wird die eigentliche Implementierung vorgestellt, die *WMC* Komponenten *K-Pool*. Hierbei wird im Rahmen des Aufbaus auch auf das zugrunde liegende Datenmodell des *K-Pool Everyplace* eingegangen. Die detaillierte Dokumentation des Plugins in Form von Klassendiagrammen und des Datenbankschemas befindet sich im Anhang (e).

5.1 Der Knowledge-Pool - Eine Einführung

Die Anwendung *K-Pool* wird am *Groupware Competence Center* der Universität Paderborn entwickelt und stellt dort das "Rückrad der Knowledge Management Infrastruktur für den Lehr-, Forschungs- und Projekmanagementbetrieb sowie den Technolgie Transfer dar"[117]. In dieser Wissensdatenbank werden so genannte *Knowledge Objects (K-Objects)* abgelegt, die zum einen aus den eigentlich (multimedialen) Inhalten und zusätzlich aus Meta Daten zur Einordnung dieser Inhalte bestehen[118].

Diese "Dokumenten und Contentmanagementplattform"[119] fokussiert dabei auf das kollaborative, also durch die Zusammenarbeit von Menschen entstehende, Wissensmanagement. Der *K-Pool* besteht dabei neben der eigentlichen Wissensdatenbank noch aus zahlreichen Zusatzmodulen, die mit der Wissensdatenbank zusammenarbeiten und zusätzliche Funktionalitäten oder Darstellungsweisen bereitstellen. Der grundsätzliche Aufbau des *K-Pools* ist aus Abbildung 5-1 ersichtlich.

---

[117] *übersetzt aus Nastansky 2005 S.2*

[118] *für eine detaillierte Betrachtung der Knowledge-Objects vergleiche Nastansky 2005 S.4 oder Hesse 2005 S.4*

[119] *vgl. Hesse 2005 S.3*

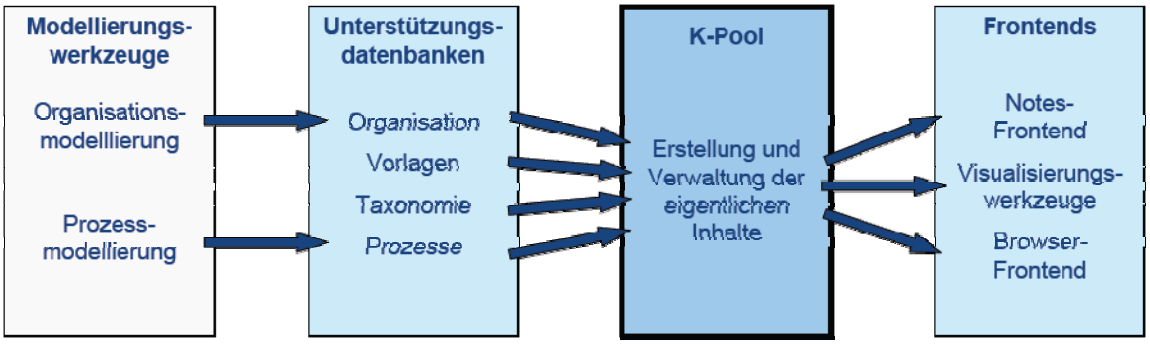


Abbildung 5-1 - Aufbau des Knowledge Pools (K-Pool)[120]

Technisch basiert der *K-Pool* auf dem Groupwaresystem *Lotus Notes/Domino* und bietet dementsprechend grundsätzlich zwei Benutzeroberflächen (Frontends). Zum einen den nativen *Lotus Notes Client* und zum anderen eine Web-Ansicht, die eine "Anbindung eines größeren Kreises von Informations Konsumenten"[121] ermöglicht. Zusätzliche Visualisierungswerkzeuge, die zunächst als Stand-Alone-Anwendungen entwickelt wurden, sind in diese Frontends eingebettet.

5.1.1 Einsatzszenario

Wie bereits angedeutet, spielt der *K-Pool* bei der Wahrnehmung der Kernaufgaben des Lehrstuhls eine entscheidende Rolle. Hier werden die Aufgaben dargestellt, die der *K-Pool* unterstützt, wobei insbesondere der Bezug von konkreten Personengruppen zu den im Abschnitt 3.1 definierten abstrakten Benutzergruppen von Wissensdatenbanken hergestellt wird.

Die abstrakte Benutzergruppe der externen Nutzer besteht grundsätzlich aus der interessierten Öffentlichkeit, wobei sich diese im Fall des *K-Pools* mehrheitlich aus Studierenden zusammensetzt. Die Web-Oberfläche des *K-Pools* ermöglicht es dieser Benutzergruppe auf öffentlich Bereiche des *K-Pools* zuzugreifen. Hierdurch werden nicht nur die Publikationen des Lehrstuhls der Öffentlichkeit zugänglich gemacht, sondern es wird zusätzlich auch die gesamte Organisation der Lehrveranstaltungen (z.B. Bereitstellung von Informationen und Arbeitsmaterialien) über diesen web-basierten Zugang zum *K-Pool* abgewickelt. Darüber hinaus übernimmt der *K-Pool* auch Funktionen

[120] *vgl. Hesse 2005 S.3*
[121] *vgl. Hesse 2005 S.10*

eines Content-Management-Systems für die Web-Seite des Lehrstuhls. So können ausgewählte Einträge durch entsprechende Funktionen des *K-Pools* auf der Web-Seite direkt sichtbar gemacht werden. Ein Beispiel für den Einsatz solcher Mechanismen ist die Bereitstellung des Inhaltes des News Bereiches der Web-Seite, durch Inhalt aus dem *K-Pool*.

Die dargestellten Einsatzszenarien für externe Benutzer konzentrieren sich, wie bei externen Benutzern üblich, auf die reine Informationsdarstellung. Der schreibende Zugriff beschränkt sich auf die Möglichkeit, Kommentare zu bestimmten *K-Objects* zu verfassen, während Zugriffe auf die eigentlichen Inhalte der *K-Objects* nur lesend erfolgen können. Im Unterschied zur abstrakten Gruppe der externen Benutzer ergibt sich im konkreten Fall tendenziell eine Verschiebung dahingehend, dass der Anteil der mehrmaligen Nutzer höher sein wird als in der abstrakten Benutzergruppe angenommen. Begründet ist dies darin, dass Studierende, die beispielsweise eine Lehrveranstaltung des Lehrstuhls besuchen, während eines bestimmten Zeitabschnitts (z.B. ein Semester) häufiger auf die Informationsseiten ihrer Veranstaltung zugreifen werden, um aktualisierte Informationen abzurufen.

Wissenschaftliche Mitarbeiter und auch Studierende, die Projekte des Lehrstuhls belegen, lassen sich der abstrakten Benutzergruppe der Wissensarbeiter zurechnen. Während Studierende, die Projekte am Lehrstuhl bearbeiten, nur mit eingeschränkten Rechten ausgestattet sind, die es ihnen erlauben, im begrenzten Umfang neue Inhalte zu erstellen, in denen z.B. die Ergebnisdokumente des eigenen Projekts abgelegt werden, verfügen Mitarbeiter des Lehrstuhls über weitergehende Rechte. Diese beinhalten beispielsweise den Zugriff auf *K-Objects* die nur intern zugänglich sind (z.B. Notenlisten oder Klausuren), aber auch das Recht, beliebige *K-Objects* zu erstellen und deren Einordnung in den Kontext (die Meta-Daten) zu erweitern und zu überarbeiten. Über diese Funktionalitäten wird auch die Content-Management-Funktionalität gesteuert, so dass wissenschaftliche Mitarbeiter auch direkten Einfluss auf die Inhalte der Web-Präsenz des Lehrstuhls nehmen können.

Als Client wird für diese Tätigkeiten sowohl von den Studierenden als auch von den wissenschaftlichen Mitarbeitern der *Lotus Notes Client* eingesetzt. Dieser ermöglicht im Vergleich zur web-basierten Oberfläche gerade im internen Netzwerk ein erheblich schnelleres Arbeiten und stellt komfortablere Suchfunktionen bereit. Darüber hinaus stellt dieser Client erweiterte Funktionalitäten zur Verfügung die beispielsweise die Einbettung von Bildern und Dateianhängen in die entsprechenden *K-Objects* ermöglichen. Somit ist

sowohl die Informationsgenerierung als auch das Informationsretrieval mit dem *Notes Client* effizienter, als über die Web-Oberfläche.

Die abstrakte Benutzergruppe der Mitarbeiter im Wissensmanagement lässt sich am Lehrstuhl nicht eindeutig einer Person oder einer Personengruppe zuordnen. Redaktionelle Aufgaben werden meist von den wissenschaftlichen Mitarbeitern durchgeführt; so trägt beispielsweise jeder Mitarbeiter die redaktionelle Verantwortung für die Inhalte, die im Rahmen von Projekten, die dieser betreut, generiert werden. Weitergehende Wissensmanagement-Aufgaben werden ohne explizit festgelegte Verantwortlichkeiten ebenfalls von wissenschaftlichen Mitarbeitern kollaborativ durchgeführt. Die Wartung des Systems wird entweder durch den Administrator oder studentische Hilfskräfte vorgenommen, während Weiterentwicklungen der Plattform und Anpassungen von Prozessen zusätzlich auch im Rahmen von Projekten erarbeitet werden. Hierbei kommt als Client neben technischen Tools der zugrunde liegenden Middleware-Plattform *Lotus Notes/Domino*, ebenfalls der *Notes Client* zum Einsatz.

5.1.2 Funktionalitäten

Die Gesamtanwendung *K-Pool* besteht aus vielen verschiedenen Komponenten die den Funktionsumfang erheblich erweitern. Aus diesem Grund erfolgt die Darstellung der Funktionen zweigeteilt. Zunächst werden die Grundfunktionen vorgestellt die für die Ablage und die Verwaltung von Informationen zentral sind. Anschließend erfolgt die Erläuterung der darüber hinaus gehenden Funktionalitäten.

5.1.2.1 Grundfunktionen

Die Grundfunktionen bestehen zusammenfassend aus der Ablage und dem Retrieval von Informationen. Die zentrale Informationseinheit hierbei ist das *K-Object*. Für den eigentlichen Inhalt des *K-Objects*, der in Form eines RichText-Feldes gespeichert wird existieren Vorlagen für unterschiedliche Anwendungszwecke (z.B. Lehrveranstaltungen). Unabhängig von der benutzten Vorlage wird das *K-Object* mit Metadaten versehen. Diese Metadaten bestehen aus einem statischen Teil und einem dynamischen Bereich, dessen Ausgestaltung durch die den *K-Pool* nutzende Organisation bestimmt werden kann, um somit Anpassbarkeit auf individuelle Bedürfnisse zu gewährleisten. Ein entscheidendes Konzept bei dieser Einordnung der Informationen mit Hilfe von Metadaten ist die Mehrfachkategorisierung. Sie sorgt dafür, dass die *K-Objects* nicht nur einem Kontext

zugeordnet sind, sondern allen relevanten Kontexten zugeordnet werden können, und somit später auch entsprechend gefunden werden. Zusätzlich zu der der Suche von *K-Objects* über die vorgenommene Einordnung durch Metadaten, bietet der *K-Pool* auch die Möglichkeit einer Volltextsuche, die sich sowohl über die abgelegten Metadaten als auch über evtl. angehängte Informationen (z.B. Dokumente) erstreckt.

5.1.2.2 Erweiterte Funktionalitäten

Neben den Grundfunktionen bietet der *K-Pool* durch zahlreiche Erweiterungen noch weitere Funktionen, die innerhalb des Lehrstuhls eingesetzt werden.

Mit Hilfe einer Prozess- und einer Organisationsdatenbank ermöglicht der *K-Pool* die Initiierung und die Abarbeitung von Workflows. Hierbei können sowohl in der Prozessdatenbank hinterlegte vordefinierte, als auch selbst erstellt ad-hoc Workflows gestartet werden. Für die Wartung der in der Prozess- und der Organisationsdatenbank hinterlegten Informationen stehen ebenfalls eigene Tools zur Verfügung. Diese Funktionalität ermöglicht beispielsweise eine koordinierte Erstellung von neuen Inhalten unter Beteiligung mehrerer Mitarbeiter. Eine weitere Funktion ist die Möglichkeit, Annotationen zu vorhandenen *K-Objects* hinzuzufügen. Auf diese Weise können sowohl zusätzlich Informationen zu vorhandenen *K-Objects* angelegt werden, ohne das Original *K-Object* zu verändern, als auch eine Diskussion zu den vorhandenen Informationen gleich im Kontext des entsprechenden *K-Objects* geführt werden.

Ein weiterer großer Bereich, in dem die eigentliche Wissensdatenbank des *K-Pools* erweitert wurde, ist der Bereich der Visualisierung der enthaltenen Informationen. Hier existieren zwei Erweiterungen, die eine völlig andere, freiere Art des Zugangs zu den im *K-Pool* abgelegten Informationen ermöglichen: Das *Hyperbolic Tree System (Star Tree)* dient dazu, Zusammenhänge die zwischen einzelnen *K-Objects* bestehen, grafisch sichtbar und auch nachvollziehbar zu machen. Das *Topic-Map System (K-Viewer)* aggregiert zusammenhängende *K-Objects* unter einem Oberbegriff (Topic) und stellt Verknüpfungen zwischen den einzelnen Oberbegriffen her. Diese Verknüpfungen werden, ähnlich wie beim *Hyperbolic Tree*, interaktiv grafisch visualisiert[122].

[122] *für weitergehende Informationen sei auf die Web-Seite des K-Discovery Projektes verwiesen*

5.2 Vorstellung der WMC Komponente K-Pool

Nach der Einführung in den vorhandenen *K-Pool* erfolgt in diesem Abschnitt die Beschreibung des implementierten Prototypen. Darüber hinaus wird in einem Exkurs auf die verschiedenen Möglichkeiten eine Komponente für den *IBM Workplace Managed Client* zu entwickeln und zu Testen eingegangen. Die Vorstellung der Komponente gliedert sich in die Darstellung des Aufbaus der Anwendung, wobei, soweit notwendig, auch auf die bereits durch ein vorhergehendes Projekt vorliegende relationale Variante des *K-Pools* eingegangen wird, und die Beschreibung der Funktionalitäten der Komponente.

5.2.1 Aufbau

Die Komponente ist eine 2-Tier-Anwendung. Das bedeutet, dass die Bereitstellung der eigentlichen Funktionalitäten der Anwendung durch die direkte Kommunikation der Komponente mit einer zentralen Datenbank erfolgt, ohne dass eine weitere Instanz als Vermittler zwischen Client und Datenbank auftritt. Eine Besonderheit ergibt sich durch die Einbindung der Komponenten in die *Workplace*-Infrastruktur. Diese führt dazu, dass der Prototyp, zusammen mit dem gesamten *WMC*, über eine weitere Serverbindung zum *Workplace*-Server verfügt, die u.a. zum Update der entwickelten Komponente genutzt wird. Diese Architektur entspricht der Rolle *Leverage* des im Abschnitt 4.2 vorgestellten *IBM* Schemas zur Einordnung von Fremdentwicklungen. Zusätzlich wird unter bestimmten Umständen noch auf einen externen *LDAP-Server* zu Authentifizierung zugegriffen. Für eine detaillierte Beschreibung, wann diese *LDAP-Server* Verbindung genutzt wird, sei auf den folgenden Abschnitt 5.2.2 und das Aktivitätendiagramm des Logins im Anhang (d) verwiesen. Den beschriebenen Aufbau der Komponente verdeutlicht Abbildung 5-2.

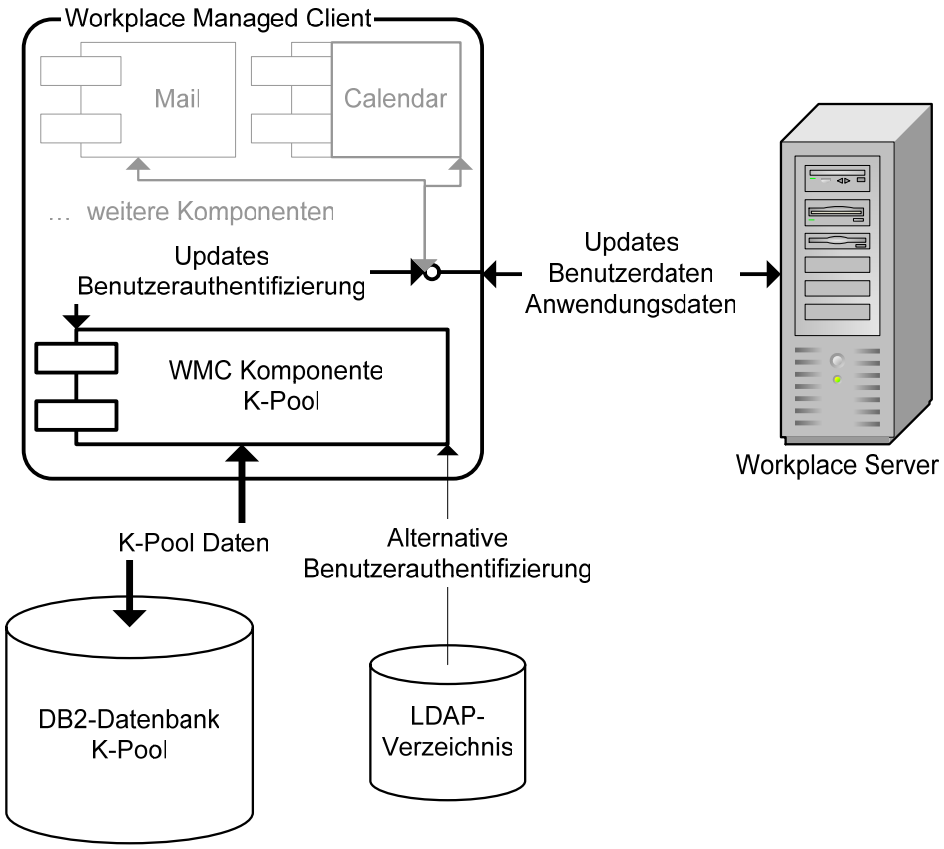


Abbildung 5-2 - Aufbau der WMC Komponente K-Pool

Aus einem vorhergehenden Projekt, in dem eine relationale web-basierte Variante des *K-Pools* entwickelt wurde, der *K-Pool Everyplace*[123], bestand bereits ein Datenmodell, welches in gleicher Form auch für die *WMC*-Komponente verwendet wurde. Dies ermöglicht es, beide Anwendungen parallel auf der identischen Datenbank ablaufen zu lassen und somit eine Interoperabilität zwischen beiden Systemen herzustellen.

Die Komponente selber ist, da es sich um ein *Eclipse*-Plugin handelt, in *Java* implementiert. Die Verbindung zur Datenbank erfolgt über entsprechende Controller Klassen, die unter Verwendung von *JDBC* mit der Datenbank kommunizieren. Zur Verringerung des Datenverkehrs zwischen der Komponente und der Datenbank werden zum einen benötigte Teile von *K-Objects* (z.B. die Detailtexte oder Anhänge) dynamisch nachgeladen und zum anderen *K-Objects* die gerade angezeigt werden lokal zwischengespeichert.

[123] *für weitere Informationen zum K-Pool Everyplace sei auf Zänger 2005 verwiesen*

Die Benutzeroberfläche der Komponente besteht aus den auch für reine *Eclipse*-Plugins typischen Komponenten. Zur Darstellung der Kategorisierungsstruktur, der Auflistung der *K-Objects* und der Vorschau werden statische Views verwendet die auf diese Weise miteinander kommunizieren können. Zur Darstellung einzelner *K-Objects* werden dynamisch neue Views generiert und mit dem anzuzeigenden *K-Object* initialisiert. Innerhalb der Views und für die weiteren Elemente der Benutzeroberfläche (z.B. Dialoge, Action Buttons, Widgets) kommen entweder direkt entsprechende Elemente des *GUI-Frameworks SWT*, oder geeignete Elemente des *Jface-Paketes* zum Einsatz. *Jface* kapselt komplexe UI-Elemente innerhalb einer Funktionseinheit und vereinfacht dadurch die Interaktion mit diesen, ohne die zugrunde liegenden *SWT-Elemente* zu verdecken. *Jface* ermöglicht u.a. eine Trennung von Modell und Darstellung (MVC-Paradigma) und stellt somit eine Abstraktionsebene für *SWT* dar. Für eine detailliert Übersicht der Anwendungsstruktur sei auf die Klassendiagramm und das Datenmodell im Anhang (e) verwiesen.

5.2.2 Funktionalität

Die *WMC*-Komponente *K-Pool* ermöglicht es, auf nahezu alle im oben angesprochenen relationalen Datenmodell deponierbaren Informationen wahlfrei zuzugreifen. Lediglich bestimmte Felder, die im *K-Pool Everyplace* zur Steuerung der Anwendung dienen, und für die eigentliche Funktionalitäten nicht benötigt werden, werden nicht angesprochen.

Für den Benutzer bedeutet dies, dass die *WMC*-Komponente *K-Pool* die im Abschnitt 5.1.2.1 beschriebenen Grundfunktionalitäten im Rahmen des relationalen *K-Pools* anbietet. Die im original *K-Pool* erreichte Flexibilität bei der Ablage von unterschiedlichsten Informationen in Rich-Text-Feldern wird in der relationalen Variante dadurch abgebildet, dass beliebige Dateien als Anhänge zu einem *K-Object* hinzugefügt werden können. Die Beschreibung und Kategorisierung der abgelegten Informationen geschieht, ähnlich wie beim Original *K-Pool*, zweigeteilt. So existieren innerhalb des *K-Objects* statische Felder, in denen entsprechende Angaben (z.B. eine Jahresangabe oder ein beschreibender Text) gemacht werden können. Darüber hinaus erfolgt die Einordnung eines *K-Objects* jedoch zusätzlich über durch den Anwender selbst bestimmbare flexible Strukturen, wie beispielsweise so genannte *Themes* oder *Keywords*.

Flexibilität war auch ein Ziel bei der Entwicklung der Benutzeroberfläche. So ist beispielsweise das Erstellen eines neuen *K-Objects* auf insgesamt drei unterschiedliche

Arten möglich. Neben Auswahl des entsprechenden Menüpunktes bzw. Buttons oder die Verwendung eines Tastaturkürzels, kann auch eine beliebige Datei per Drag and Drop auf ein Element der selbst erstellten Ordnungsstruktur (z.B. ein bestimmtes *Theme*) gezogen werden. Dies hat den Vorteil, dass im dabei entstehenden *K-Object* sowohl die verwendete Datei automatisch angehängt wird, als auch bereits eine Einordnung des *K-Objects* zum entsprechenden Element vorgenommen wurde. Drag and Drop kann ebenso benutzt werden um vorhandenen *K-Objects* anderen Elementen zuzuordnen, oder ganze Strukturen von Kategorisierungen neu zu ordnen.

Die Auslegung als Rich-Client-Anwendung zeigt sich gleich an mehreren Stellen. So werden, da ein Großteil der Logik durch den Client ausgeführt wird, Verarbeitungskapazitäten der Client-Hardware genutzt. Dies zeigt sich bei der Validierung, der Möglichkeit Daten lokal zu sortieren, oder auch bei der Skalierung von Bildern für die *K-Objects*.

Die gute Eingliederung der Anwendung in das Look-and-Feel des Betriebssystems wird im wesentlichen durch das verwendete *RCP-Framework* und die damit einhergehende Nutzung von *SWT* zur Gestaltung der Oberfläche erreicht. Zusätzlich wurde im Bereich des *Feedback* Wert darauf gelegt, den Benutzer (bei kürzeren Aufgaben durch die Veränderung des Mauszeigers, bei längeren Wartezeiten (z.B. der Download eines Dateianhangs) durch entsprechende Statusdialoge) über den aktuellen Zustand der Anwendung informiert zu halten (vgl. Abbildung 5-3).

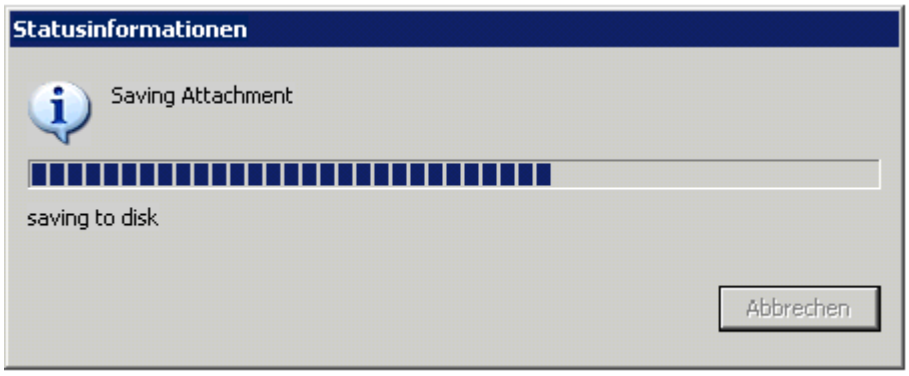


Abbildung 5-3 - Statusanzeige beim Download von Attachments

Das Sicherheitskonzept der Komponente entspricht dem bereits im *K-Pool Everyplace* eingesetzten Konzepts, das fünf Rollen zwischen Reader und Admin unterscheidet. In jedem *K-Object* wird die für die Sichtbarkeit dieses Objects mindestens erforderliche Rolle

explizit festgelegt. Zusätzlich kann den in einem *K-Object* hinterlegten Dateianhängen ebenfalls eine mindestens erforderliche Rolle zugewiesen werden. Auf diese Weise können besonders sensible Informationen innerhalb eines *K-Object* verwaltet werden. Für eine Übersicht über die verschiedenen Rollen und deren Rechte sei auf den Anhang (c) verwiesen.

Neben den angesprochenen Grundfunktionalitäten ermöglicht die *WMC* Komponente *K-Pool* auch die Benutzerverwaltung und die Administration der gesamten Anwendung. Die Administration wird durch das Anlegen und Aktivieren von so genannten Profilen vorgenommen. Diese Profile ermöglichen hauptsächlich die Konfiguration der Web-Anwendung *K-Pool Everyplace*. Für die vorliegende Komponente wird aus dem Profil lediglich die Einstellung für einen evtl. verwendeten *LDAP-Server* benutzt. Trotzdem ist das Anlegen eines vollständigen Profils sinnvoll, da dieses bei Verwendung einer gemeinsamen Datenbank auch durch den *K-Pool Everyplace* genutzt werden kann.

Die Benutzerverwaltung erfolgt durch Eingabe des entsprechenden Benutzers und die Vergabe einer entsprechenden Rolle innerhalb der *Access Control List (ACL)* des *K-Pools*. Die eigentliche Authentifizierung des Benutzers erfolgt, um ein *Single Sign On (SSO)* zu ermöglichen, entweder durch den *Workplace*-Server oder einen externen *LDAP-Server*. Die Komponente versucht zunächst, den aktuellen *Workplace*-Benutzer an den *K-Pool* anzumelden. Dies gelingt dann, wenn der aktuelle Benutzer als *K-Pool*-Benutzer in der *ACL* des *K-Pools* eingetragen ist; dieser bekommt dann die in der *ACL* festgelegte Rolle zugewiesen. Andernfalls fordert die Komponente den Benutzer auf, einen anderen Benutzernamen und das entsprechende Passwort einzugeben. Diese Angaben werden daraufhin mit dem im Profil der Anwendung eingestellten *LDAP-Server* abgeglichen. Gelingt dies, erfolgt ebenso die Kontrolle ob dieser Benutzer in der *ACL* des *K-Pools* eingetragen ist. Alternativ besteht die Möglichkeit, den *K-Pool* als anonymer Benutzer mit minimalen Zugriffsrechten zu nutzen. Ein Aktivitätendiagramm, welches diesen Login Vorgang illustriert, befindet sich im Anhang (d).

5.2.3 Exkurs: Vom Eclipse Plugin zur Workplace Komponente

In diesem Exkurs werden die Unterschiede und Besonderheiten, die bei der Entwicklung einer *WMC*-Komponente im Vergleich zur Entwicklung einer reinen *RCP*-Anwendung auftreten aufgezeigt. So wurden im Rahmen dieser prototypischen Implementierung, vor allem dadurch bedingt, dass technische Voraussetzungen verändert oder erweitert wurden,

einige Teile der Anwendung mehrfach programmiert, um den geänderten Rahmenbedingungen zu entsprechen. Aufbauend auf diesen Erfahrungen werden Empfehlungen gegeben in welcher Weise die Implementierung einer *WMC*-Komponente erfolgen sollte, um den Entwicklungsaufwand möglichst gering zu halten.

Die zentrale Frage bei der Entwicklung einer *WMC*-Komponente betrifft die zu verwendende Testumgebung. Hier ergeben sich bei Verwendung von *Eclipse* als Entwicklungsumgebung für die zu erstellende Komponente drei unterschiedliche Möglichkeiten, die im Folgenden kurz dargestellt werden.

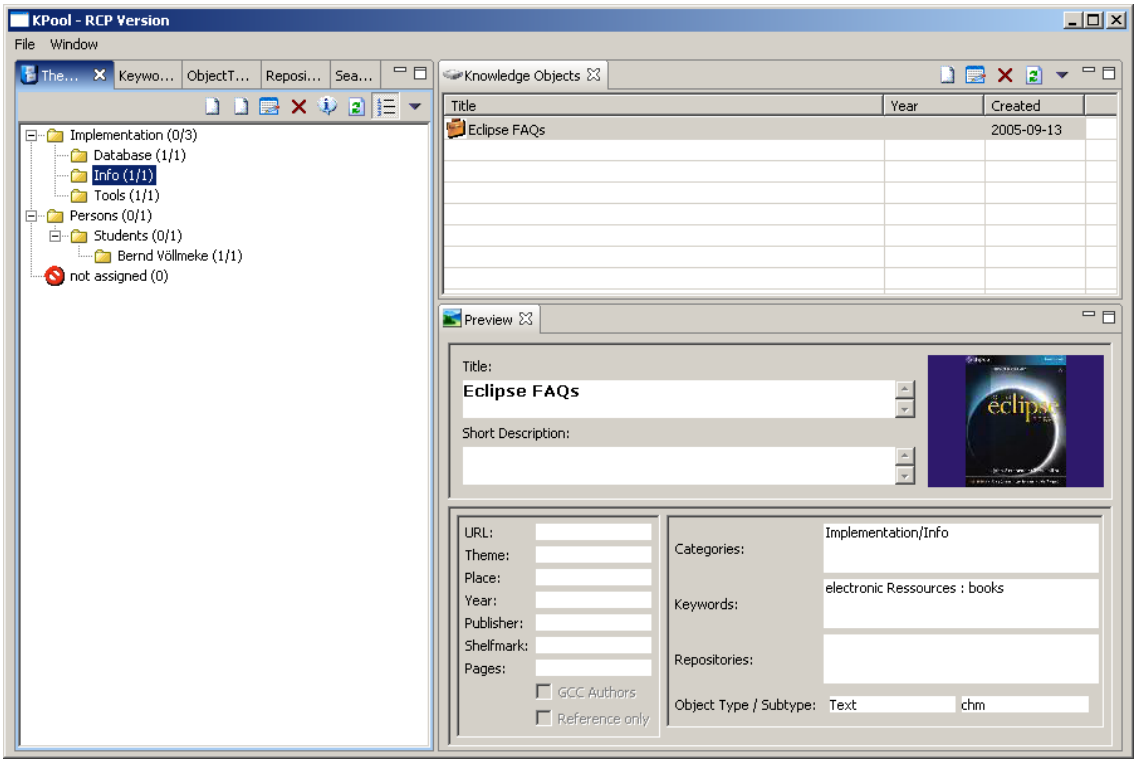


Abbildung 5-4 - Screenshot: Test als RCP Anwendung

Eclipse RCP-Anwendung

Die Nutzung der in der *Eclipse* Version 3.1 vorhandenen und im Tutorial von Ed Burnette[124] erklärten Funktionen zur Erstellung eines eigenständigen *RCP*-Projekts stellt die einfachste Möglichkeit dar, einen Rahmen für eine zu programmierende *WMC*-Komponente bereitzustellen. Neben dem geringen Erstellungsaufwand ermöglicht diese Konfiguration auch ein sehr schnelles Testen der Anwendung, da der Startvorgang für eine

[124] *vgl. Burnette 2005*

solche Applikation in wenigen Sekunden abgeschlossen ist. Nachteilig ist jedoch, dass, da es sich hierbei um eine reine *RCP*-Anwendung handelt (vgl. Abbildung 5-4), Funktionen, die der *WMC* zusätzlich zum *RCP-Framework* bereitstellt, in einer solchen Testumgebung nicht genutzt werden können.

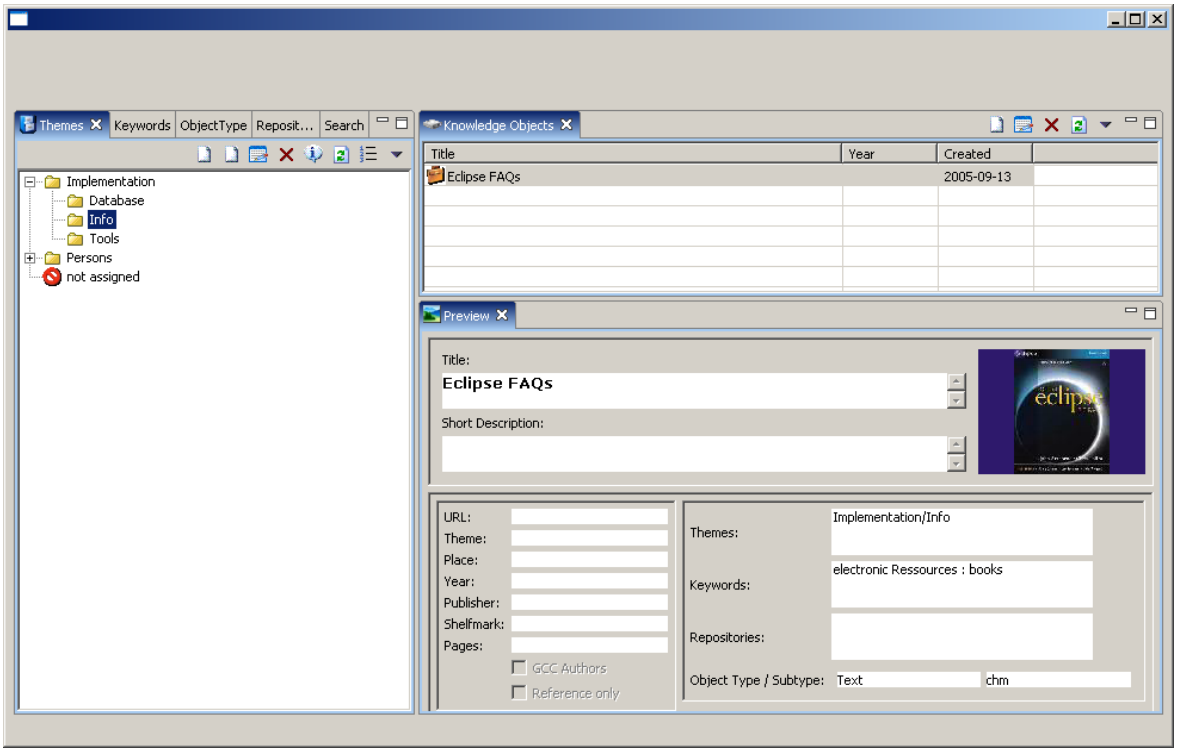


Abbildung 5-5 - Screenshot: Test mit eigener Personality

WMC-Anwendung mit eigener Personality

Die zweite Möglichkeit, eine Testumgebung für die selbst entwickelte Komponente bereitzustellen, besteht darin, diese im *WMC* mit Hilfe einer eigenen Personality zu starten(vgl. Abbildung 5-5). Dieses Vorgehen ist in der Dokumentation der *Lotus-Workplace-API* beschrieben[125]. Die selbst erstellte Personality ist dabei das *Workplace*-Equivalent zu einer *Eclipse*-Perspective. In ihr wird festgelegt, welche Komponenten beim Start des Clients angezeigt werden. Um diese Methode zu nutzen, muss bereits ein *WMC* auf dem Testsystem installiert sein, und darüber hinaus die Entwicklungsumgebung zur Nutzung dieser *WMC*-Installation konfiguriert sein[126].

[125] *vgl. IBM 2005 S.39ff.*
[126] *vgl. IBM 2005 S.7ff.*

Da ein installierter *WMC* als Runtime-Umgebung für die selbst erstellte Komponente benutzt wird, ist es mit Hilfe dieser Konfiguration möglich, auf die API des *WMC*s zuzugreifen, und somit erweiterte Funktionen des Clients zu nutzen. Lediglich Funktionen, die vom Login des *WMC* abhängig sind können nicht getestet werden, da keine Benutzeranmeldung beim Start erfolgt. Das Look-And-Feel der Komponente entspricht bei dieser Möglichkeit weitgehend dem der *RCP*-Anwendung. Der Startvorgang ist etwas langsamer als eine reine *RCP*-Anwendung, da Teile des *WMC* beim Testen der Anwendung ebenfalls gestartet werden.

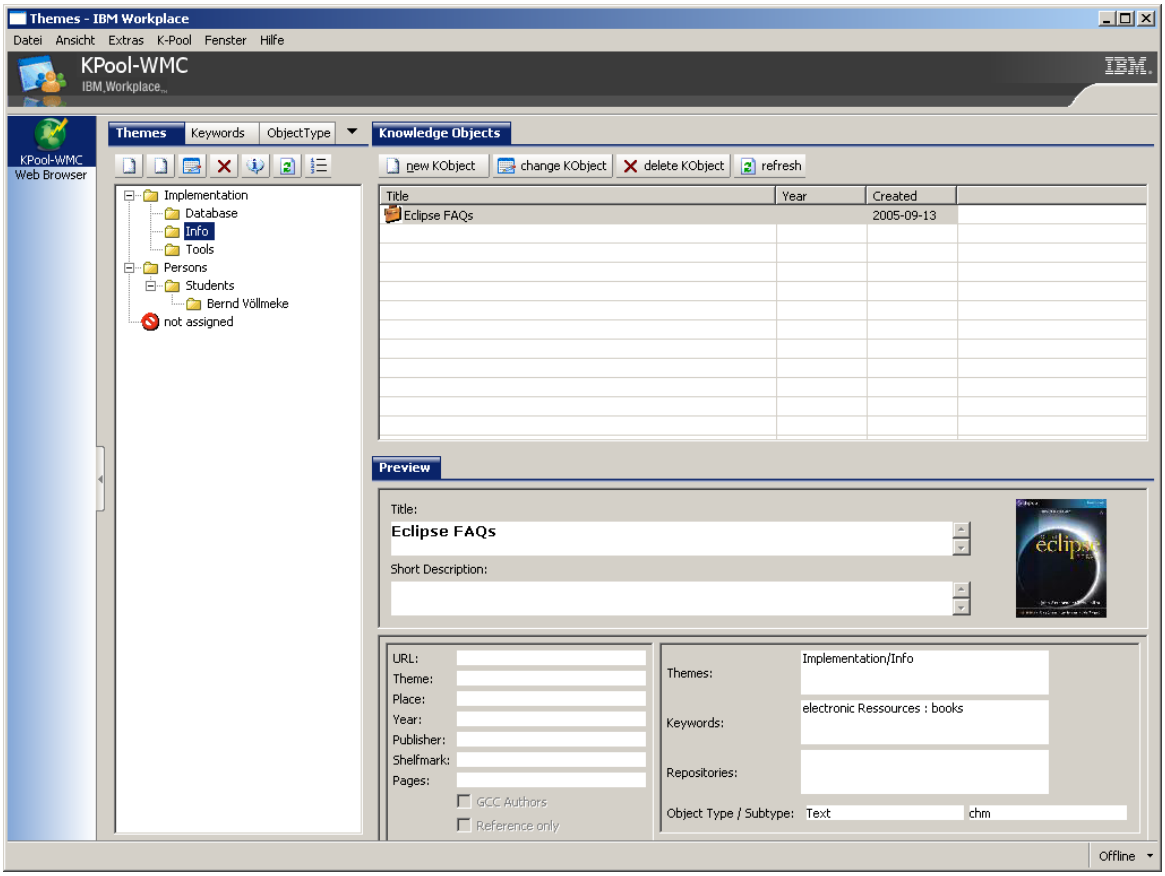


Abbildung 5-6 - Screenshot: Test unter Nutzung des WMC Developer Toolkits

Nutzung des WMC Developer Toolkits

Oben genanntes Toolkit ist eine von *IBM* entwickelte Erweiterung der *Eclipse*-Entwicklungsumgebung. Eine Einführung zur Nutzung des Toolkits findet sich ebenfalls auf der *IBM* Web-Seite[127]. Das Toolkit übernimmt weitgehend automatisiert die Konfiguration einer Testumgebung (vgl. Abbildung 5-6) und erstellt über einen

[127] *vgl. Sewell 2005*

Assistenten ein konfigurierbares Skelett einer *WMC*-Komponente, auf dem man bei der weiteren Entwicklung aufbauen kann. Darüber hinaus enthält das Toolkit einen Export-Assistenten, der die für die Installation der Komponente notwendigen Dateien automatisch zusammenstellt. Eine Vorraussetzung für den Einsatz des Toolkits ist, ebenso wie bei der zuletzt vorgestellten Variante, ein installierter *WMC*.

Neben dem Vorteil der vereinfachten Konfiguration startet das Toolkit die selbst erstellte Komponente als vollwertige *WMC*-Komponente. Das bedeutet, dass sowohl das Login des Clients vorgenommen wird, als auch dass Look-And-Feel des *WMC* für die neue Komponente verwendet wird. Hiermit stellt diese Testumgebung die umfassendsten Möglichkeiten bereit, da sowohl *Workplace*-Funktionen, die ein Login erfordern, getestet werden können, als auch die optische Integration der Komponente in den *WMC* überprüft werden kann. Der Export-Assistent eliminiert zudem die Notwendigkeit, aufwändige Konfigurationsdateien von Hand zu erstellen.

Problematisch an dieser Testumgebung sind allerdings, neben einigen noch enthaltenen Fehlern[128] im Toolkit, vor allem Probleme, die durch die Verwendung des *WMC*-Designs auftreten, sowie die lange Startzeit der Testumgebung. Das *WMC*-spezifische Design blendet bestimmte Elemente, die in den beiden zuvor genannten Testumgebungen sichtbar sind, aus. Hierzu zählen beispielsweise die Symbole, die in den Registerkarten von Views verwendet werden, aber auch die so genannte Button-Bar der Views, in der die Aktionen eines Views angeboten werden. Letzteres hat zur Folge, dass die schon programmierten Aktionen eines Views, damit diese auch in dieser spätern Zielumgebung zu sehen sind, zu einer *Workplace*-spezifischen Button-Bar hinzugefügt werden müssen. Die lange Startzeit der Testumgebung ist durch das vollständige Starten des *WMC* bedingt, zusätzlich muss bei jedem Start die manuelle Eingabe des entsprechenden *Workplace*-Passworts erfolgen.

Die Vor- und Nachteile der vorgestellten Testumgebungen führen zu der Empfehlung, das grundlegende Design der Komponente, also beispielsweise die Anordnung von GUI-Elementen in eigenen Views, zunächst in Form einer reinen *RCP*-Anwendung durchzuführen. Da gerade bei der Gestaltung der Benutzeroberfläche sehr kurze Testzyklen typisch sind, bietet sich diese Testumgebung aufgrund ihrer kurzen Startzeit an. Während der weiteren Entwicklung sollte das Entwicklungsprojekt dann erst, wenn die

[128] *so werden in der aktuellen Version 1.0.0 noch bei jedem Start der Anwendung bestimmte Daten aus der Laufzeitumgebung gelöscht (u.a. die für die Mementos verwendeten Daten)*

entsprechenden speziellen Funktionen des *WMC* benötigt werden, auf die jeweils benötigte Testumgebung umgestellt werden. Wann diese Umstellungen erfolgen sollten, hängt somit maßgeblich von dem Grad der Integration der Komponente in die vom *WMC* angebotenen Dienste ab. Obwohl bei dieser Methode zusätzlicher Aufwand durch die Umstellung des Projektes auf die jeweils neue Testumgebung entsteht, kann durch dieses Vorgehen der Zeitbedarf für die gesamte Entwicklung minimiert werden, da ein großer Anteil der für die Entwicklung notwendigen Testläufe in vergleichsweise schnell startenden Testumgebungen vorgenommen werden kann. Darüber hinaus sollte bei der Entwicklung schon zu Anfang berücksichtigt werden, dass bestimmte Funktionen in der endgültigen *Workplace*-Komponente nicht genutzt werden können, (z.B. Symbole für Views) und dementsprechend kein Entwicklungsaufwand auf diese Funktionen verwendet wird.

6 Fazit

Die vorliegende Diplomarbeit führt den Leser in die Grundlagen der klassischen Client-Konzepte, dem Thin-Client (speziell der Browser als verbreitete Variante) auf der einen, und dem Thick-Client auf der anderen Seite ein. Die Verbindung zum späteren Einsatzgebiet des Prototypen wird durch die Darstellung des Themenbereichs der wissensintensiven Arbeitsumgebungen und Wissenssammlungen - als ein Konzept im Wissensmanagement - gegeben. Kernaussage ist in diesem Bereich, neben der festzustellenden wachsenden Bedeutung von wissensintensiver Arbeit, die Abgrenzung von komplexen wissensintensiven Prozessen zu aus anderen Arbeitsbereichen bekannten Routinetätigkeiten. Die Ausübung einer wissensintensiven Tätigkeit erfordert nicht nur eine gute formale Ausbildung der Wissensarbeiter, sondern bringt auch Unterschiede in der Nutzung eines unterstützenden IT-Systems mit sich. Aufbauend auf diesen Erkenntnissen werden drei unterschiedliche Benutzergruppen (Externe Nutzer, Wissensarbeiter, Mitarbeiter im Wissensmanagement) für Wissensdatenbanken identifiziert, anhand derer die Analyse der Anforderungen an die für den Zugriff auf die Wissensdatenbank zu benutzende Software erfolgt.

Weiterhin werden bestehende Konzepte zur Bewertung von Software vorgestellt. Auf der einen Seite steht die Usability einer Software; dieser Oberbegriff wird detailliert auf einzelne Komponenten herunter gebrochen und darauf aufbauend Grundsätze für das Design einer Anwendung entwickelt. Dem vorgestellten Ansatz, den aus Usability entstehenden Nutzen monetär zu bewerten, stehen verschiedene Autoren kritisch gegenüber, da die wichtigsten Inputfaktoren für die verwendeten Methoden auf letztendlich subjektiven Schätzungen basieren, und des Weiteren die Rahmenbedingungen für den Einsatzes der Software vernachlässigt werden. Im konkreten Szenario der wissensintensiven Arbeitsumgebungen treten gerade letztgenannte Probleme verstärkt auf. Dennoch ist es unstrittig, dass eine gute Usability von Software einen, wenn auch in wissensintensiven Arbeitsumgebungen nur indirekten, positiven Einfluss auf die Produktivität der Mitarbeiter hat.

Auf der anderen Seite steht mit den *TCO* ein grundlegend anderer Ansatz für die Bewertung von IT-Systemen zu Verfügung. Im Rahmen der Arbeit wird die Konzeption eines *TCO*-Modells am Beispiel vorgestellt und auf Variationen verschiedener Anbieter hingewiesen. Obwohl die strukturierte Erfassung von Kosten für eine IT-Infrastruktur

entscheidende Steuerungsinformation für das Management dieser Infrastruktur geben kann, zeigte sich, dass eine Vergleichbarkeit von TCO-Berechnungen nur sehr begrenzt gegeben ist. Dies liegt sowohl daran, dass es sich bei der TCO-Berechnung nicht um ein standardisiertes Verfahren handelt und verschiedene Modelle daher z.T. deutliche Unterschiede aufweisen, als auch daran, dass sich durch unterschiedliche Strukturen der berechnenden Organisation (z.B. Branche) Unterschiede ergeben. Neben dieser fehlenden Vergleichbarkeit ist ein weiterer Hauptkritikpunkt der TCO-Analyse die Ausblendung von Nutzenaspekten. Trotz dieser Schwächen ist das TCO-Modell geeignet, im weiteren Verlauf der Arbeit dazu genutzt zu werden, die Ansatzpunkte von Stärken und Schwächen der unterschiedlichen Client-Arten im Bereich der TCO qualitativ zu illustrieren.

Die, unabhängig vom Szenario und den gebildeten Benutzergruppen, vorgenommene Bewertung der beiden grundsätzlichen Client-Arten Thin- und Thick-Client zeigt, warum die Vorteile von Thick-Clients hauptsächlich im Usability-Umfeld und die Vorteile von Thin-Clients vor allem im Bereich der TCO liegen. In einem weiteren Schritt wird es durch die Verknüpfung der Usability und TCO-Aspekte mit den für das Szenario gebildeten Benutzergruppen möglich, die Anforderungen der Benutzergruppen an die einzelnen Aspekte der Usability und der TCO qualitativ deutlich zu machen. Hier zeigen sich zum Teil erhebliche Unterschiede in den Anforderungen, die sich beispielsweise aus der unterschiedlichen Nutzungsfrequenz und Nutzungsdauer ergeben, so dass Lernprozesse ungleich gewichtet werden. Zusätzlich sollten bei den externen Benutzern auch weitere Ziele, die eine Organisation mit der Veröffentlichung von Informationen der Wissensdatenbank verfolgt, einbezogen werden, da die TCO im Client-Bereich nicht relevant ist. Im Bereich der Wissensarbeiter ist die starke Beachtung von Usability-Aspekten, welche die Produktivität betreffen, bei gleichzeitiger hoher Bedeutung der TCO hervorzuheben. Mitarbeiter im Wissensmanagement haben in diesem Bereich der Usability im Vergleich zu den Wissensarbeitern noch höhere Anforderungen, allerdings nimmt hier die Bedeutung der TCO aufgrund der relativ kleineren Benutzergruppe wieder ab.

Bevor nun auf Grundlage der festgestellten Anforderungen eine Zuordnung von Benutzergruppen zu Client-Arten vorgenommen wird, erfolgt die Vorstellung einer neuen evolutionären Client-Art, die Rich-Clients der zweiten Generation (RC^2). Das Ziel dieser Client-Art ist die Verbindung der Vorteile von Thin- und Thick-Client in einem Konzept. Die Arbeit beschreibt drei grundlegend unterschiedliche technische Realisierungsansätze. Die erweiterten Browser-Anwendungen als Weiterentwicklung der klassischen Browser-Anwendungen und die plugin-basierten Konzepte unter Nutzung spezieller

Laufzeitumgebungen, die innerhalb des Browser ausgeführt werden, beinhalten den Web-Browser als Bestandteil ihrer Architektur. Dem entgegen stehen die Stand-Alone Lösungen, deren Wurzeln im Bereich von klassischen Programmiersprachen zu finden sind, und die nun durch zusätzliche Mechanismen einfacher zu installieren und zu warten sind. Die Eigenschaften der zuletzt genannten Clients werden sowohl allgemein dargestellt als auch die Vorteile, die speziell für den Bereich der Wissensdatenbanken relevant sind dargelegt. Als Hauptvorteil ist hierbei die einfache Installation und Wartung zu nennen, die gerade auch im Bereich der Wissensdatenbanken aufgrund der großen, evtl. räumlich stark verteilten Benutzergruppe der Wissensarbeiter zum Tragen kommt.

Die durch die Einführung der RC² auf drei Ausprägungen angewachsenen Client-Arten werden zum Abschluss des theoretischen Teils der Arbeit, als Zusammenfassung der gewonnenen Erkenntnisse, den identifizierten Benutzergruppen grundsätzlich zugeordnet. Hierbei werden die Überlappungsbereiche zwischen den verschiedenen Client-Arten dargestellt und die Einflussfaktoren, welche die Entscheidung der Client-Art im Einzelfall bestimmen aufgedeckt, so dass die dargestellte Zuordnung auf unterschiedliche Szenarien übertragen werden kann.

Im praktischen Teil werden zunächst die eingesetzten Technologien, das *Eclipse RCP-Framework* und der darauf aufbauende *Workplace Managed Client* erläutert. Die nach dem Vorbild der bereits mit Groupware-Technologie implementierten Wissensdatenbank *K-Pool* entwickelte *WMC*-Komponente bietet die Grundfunktionen des Originals an und beweist die praktische Funktionsfähigkeit des Prinzips des Stand-Alone RC². Auch wenn die Funktion der Offline-Fähigkeit für diesen ersten Prototypen aufgrund (noch) fehlender Standardisierung der verwendeten Plattform in diesem Bereich nicht realisiert werden konnte, so ist doch insbesondere die server-basierte Verteilung und Administration der Komponente als ein herausragendes Merkmal der RC² erfolgreich gezeigt worden. Darüber hinaus bietet diese Diplomarbeit Anknüpfungspunkte für Folgeprojekte, welche die Weiterentwicklung der im Rahmen dieser Arbeit implementierten Komponente, aber auch den durch die gleiche Datenbasis verbundenen *K-Pool Everyplace* betreffen[129]. Schließlich

[129] *eine Auflistung zukünftiger Entwicklungspotentiale findet sich im Anhang (vgl. Tabelle „Entwicklungspotentiale“ (g))*

wurden wertvolle Erfahrungen für zukünftige Projekte, die eine Komponente für die *Workplace*-Plattform entwickeln, in Form des Exkurses zur Komponentenentwicklung (vgl. Abschnitt 5.2.3) festgehalten.

7 Literaturverzeichnis

[Alvesson 2004]
Alvesson, Mats: Knowledge Work and Knowledge-Intensive Firms. Oxford University Press, New York 2004.

[Bisconti, McKinney 2005]
Bisconti, Ken; McKinney, Sue (2005): IBM Workplace: Overview and Strategy, IBM, 2005; Aus: http://fbi.zhwin.ch/swp/04-Companies/STR103.pdf am 26.09.2005.

[DIN ISO 9241-10 1996]
Deutsches Institut für Normung e.V. (1996): DIN ISO EN 9241 Teil 10 - Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten - Teil 10: Grundsätze der Dialoggestaltung.. Berlin 1996.

[DIN ISO 9241-11 1998]
Deutsches Institut für Normung e.V. (1998): DIN ISO EN 9241 Teil 11 - Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten: Anforderungen an die Gebrauchstauglichkeit. Berlin 1998.

[IBM 2005]
IBM Workplace Client Technology API Toolkit User Guide, IBM, 2005; Aus: http://doc.notes.net/uafiles.nsf/docs/wcs251api/$File/lwpapi251_wct_ug.pdf. am 27.09.2005.

[Biskamp 1998]
Biskamp, Stefan (1998): Gartner Group warnt vor TCO-Analysen. In: Information Week, 91 (1998), S. 10.

[Broy 1999]
Broy, Manfred (Hrsg.): Verein Deutscher Ingenieure (1999): VDI-Lexikon Informatik und Informationstechnik. Springer, Berlin 1999.

[BSI 2000]
Bundesamt für Sicherheit in der Informationstechnik (2000): Empfehlungen zum Schutz vor verteilten Denial of Service-Angriffen im Internet (Version1.1a). Bonn 2000.

[Burnette 2005]
Burnette, Ed (2005): Rich Client Tutorial Part 1. 2005; Aus: http://www.eclipse.org/articles/Article-RCP-1/tutorial1.html am 29.09.2005.

[Cappuccio et al. 1996]
Cappuccio, David J. ; Kirwin, Bill ; Pawlick, Lock ; Namasivayam, Siva (1996): Total Cost of Ownership: Reducing PC/LAN-Costs in the Enterprise. GartnerGroup, Stamford 1996.

[Computerworld 2005]
Computerworld (1996): Lexikon der aktuellen Fachbegriffe aus Informatik und Telekommunikation. vdf, Hochsch.-Verl. an der ETH, Zürich 2005.

[Conklin 1996]
Conklin, Jeff E. (1996): Designing organizational memory: Preserving intellectual assets in a knowledge economy. 1996; Aus: http://www.cognexus.org/dom.pdf. am26.08.2005.

[Daum 2004]
Daum, Berthold (2004): Java-Entwicklung mit Eclipse 3. dPunkt Verlag, Heidelberg 2004.

[Davenport u. Prusak 1998a]
Davenport, Thomas H.; Prusak, Laurence (1998): Wenn ihr Unternehmen wüsste was es alles weiß. Moderne Industrie, Landsberg/Lech 1998.

[Davenport u. Prusak 1998b]
Davenport, Thomas H.; Prusak, Laurence (1998): Working Knowledge: how organizations manage what they know. Harvard Business School Press, Boston, Mass 1998.

[Diezmann 2002]
Diezmann, Tanja (2002): Navigation und Usability. In:Usability – nutzerfreundliches Web-Design. Hrsg.: Beier, Markus; von Gizycki, Victoria. Springer, Heidelberg 2002.

[Däumler 1989]
Däumler, Klaus-Dieter (1989): Grundlagen der Investitions- und Wirtschaftlichkeitsrechnung. Verlag Neue Wirtschaftsbriefe, Herne, Berlin 1989.

[Edgar 2004]
Edgar, Nick (2004): Overview of the Generic Workbench. 2004; Aus: http://www.eclipse.org/rcp/EclipseCon2004/RCP%20UI.ppt am 27.09.2005.

[Fraunhofer 2005]
Fraunhofer (2005): Klassische Produktionsfaktoren (Glossareintrag). 2005; Aus: http://www.iff.fraunhofer.de/de/glossar/glossare.php?letter=E am 29.09.2005.

[Gamma, Beck 2003]
Gamma, Erich; Beck, Kent (2003): Contributing to Eclipse: Principles, Patterns, and Plug-Ins. Addison Wesley, Boston 2003.

[Gerhardt, Wege 2004]
Gerhardt, Frank; Wege, Christian : Eclipse als Basis für Rich-Client-Anwendungen - Neuer Reichtum. In: iX, 7 (2004), S. 44–47.

[Greulich 2003]
Greulich, Walter (Hrsg.) (2003): Der Brockhaus Computer und Informationstechnologie. Brockhaus Verlag, Mannheim 2003.

[Groh 2000]
Groh, Gerald (2000): TCO - Diskussion und Anbieterpositionierung. In: Forum TCO: Total Cost of Ownership. Hrsg: Bullinger, Hans-Jörg. IAO, Stuttgart 2000.

[Hartlieb 2002]
Hartlieb, Erich (2002): Wissenslogistik - Effektives und effizientes Management von Wissensressourcen. Wiesbaden : DUV, 2002.

[Heidloff, Rao 2005]
Heidloff, Niklas; Rao, Sirinivas (2005): Introduction to Developing Applications on IBM Workplace Client Technology. IBM, 2005; Aus: http://www-10.lotus.com/ldd/sandbox.nsf/f72790492fc99f0e852567ec006aa062/5ac7d643c765cfdb85256fa1004bdd0d?OpenDocument am 26.09.2005.

[Heinhold 1989]
Heinhold, Michael (1989): Investitionsrechnung. Oldenbourg, München 1989.

[Hesse 2005]
Hesse, Bernd (2005): GCC K-Pool als Knowledge Management Plattform im Intra- und Extranet. GCC, 2005; Aus: http://gcc.uni-paderborn.de/www/WI/WI2/wi2_lit.nsf/0/529e572a0aa624d8c125701f006aaeff/$FILE/DNUG%20University%20Meeting%202005%20-%20GCC%20K-Pool.pdf am 27.09.2005.

[Hirschmeier 2002]
Hirschmeier, Markus (2002): Business Engineering - IT Economics.Erlangen 2002; Aus: http://www.wi3.uni-erlangen.de/lehre/lv/ws2002/BE/BE-ws02-11.ppt am: 26.09.2005.

[Jordan 1998]
Jordan, Patrick W. (1998): An Inroduction to Usability. Taylor& Francis, London, Philidelphia 1998.

[Kanter 1998]
Kanter, Joel P. (1998): Understanding Thin-Client/Server Computing. Microsoft Press, Washington 1998.

[Karner 2005]
Karner, Herbert (2005): Was kostet Informationstechnologie. Wien 2005; Aus: http://www.eday.at/vortraege/Julius_Raab_Saal/01/karner_herbert.pdf am 30.08.2005.

[Kraenzel 2004]
Kraenzel, Carl (2004): Workplace Client Technology (Rich Client Edition) ISV integration guide. IBM, 2004; Aus: http://www.redbooks.ibm.com/redpapers/pdfs/redp3883.pdf am 26.09.2005.

[Kumar 2004]
Kumar, Suman (2004): Gmail: Usability Issues. 2004; Aus: http://www.sumankumar.com/usability/2004/05/gmail-usability-issues.html am 27.09.2005.

[Li 2003]
Li, Elma (2004): Total Cost of Ownership in Local Area Networks. 2003; Aus: http://www.elmaweb.com/www/doc/TCO_LAN.pdf am 26.09.2005.

[Lindgaard 1994]
Lindgaard, Gitte (1994): Usability Testing And System Evaluation. Chapman & Hall, London 1994.

[Lütge 2002]
Lütge, Holger (2002): Total Cost of Ownership - Ein Überblick. Berlin 2002; Auf: http://www.duett.de/it-service/wissen/int_whitepapers/tco_duett.pdf am 26.09.2005.

[Maedche 2002]
Maedche. Alexander(2002): Semantikbasiertes Wissensmanagement: Neue Wege für das Management von Wissenssammlungen. In: Wissensmanagement. Hrsg: Bellman, M.; Sommerlatte T.; Krzcmar H., Symposium Verlag, 2002; Aus: http://www.fzi.de/KCMS/kcms_file.php?action=link&id=73 am 29.09.2005.

[Mayhew, Mantei 1994]
Mayhew, Deborah J.; Mantei, Marilyn(1994): A Basic Framework for Cost-Justifying Usability. In: Cost-Justifying Usability. Hrsg: Mayhew, Deborah J.; Mantei, Marilyn, Academic Press, London 1994, S. 9–44.

[McDermott 1999]
McDermott, Richard (1999): Knowing is a Human Act: How Information Technology Inspired But Cannot Deliver Knowledge Management. In: California Management Review, 41-4 (1999), S. 103–117.

[Meyer 2003]
Meyer, John (2003): Return of the Rich Clients. Giga Information Group, 2003; Aus: http://msdn.microsoft.com/netframework/programming/winforms/richclient.aspx am 27.09.2005.

[Miedl 2003]
Miedl, Wolfgang (2003): Die Renaissance des Rich Client. In: Computerwoche Online, November 2003; Aus: http://www.computerwoche.de/index.cfm?pageid=255&artid=55380&type=detail&category=258 am 29.09.2005.

[Nastansky 2005]
Nastansky, Ludwig (2005): K-Pool: A Process-driven Knowledge Management System for Contextual Collaboration Spanning Intranet to Internet. GCC, 2005; Aus: http://pbfb5www.uni-paderborn.de/www/WI/WI2/wi2_lit.nsf/38d85b5666bcdf3dc1256f50004de5ea/a9311fa5218381cfc1256fa2003fb561/$FILE/IADIS-05_LN-UPB.pdf. am 27.09.2005.

[Nieh et al. 2000]
Nieh, Jason; Yang, S. J.; Novik, Naomi (2000): A Comparison of Thin-Client ComputingArchitectures. Columbia University.Version, New York 2000; Aus: http://www.ncl.cs.columbia.edu/publications/cucs-022-00.pdf. am 29.09.2005.

[Nielsen 1993]
Nielsen, Jakob(1993): Usability Engineering. Academic Press, Boston 1993.

[Nielsen, Mack 1994]
Nielsen, Jakob; Mack, Robert L. (1994): Usability Inspection Methods. Wiley, New York 1994.

[Oakley et al. 2000]
Oakley, Ian; McGee, Marilyn R.; Brewster, Stephen; Gray, Philip (2000): Putting the feel in 'look and feel'. In: CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems. ACM Press, New York 2000, S. 415–422.

[Pilgrim 2004]
Pilgrim, Mark (2004): Gmail-Accessibility. 2004; Aus: http://diveintomark.org/archives/2004/04/10/gmail-accessibility am 27.09.2005.

[Potthoff 1998]
Potthoff, Ingo (1998): Kosten und Nutzen der Informationsverarbeitung : Analyse und Beurteilung von Investitionsentscheidungen. Deutscher Unversitätsverlag, Wiesbaden 1998.

[Riepl 1998]
Riepl, Ludwig (1998): TCO vs. ROI. In: Information Management 2 (1998), S. 7–12.

[Rosenberg 2004]
Rosenberg, Daniel (2004): The myths of usability ROI.. In: Interactions 11 (2004) October/November, Nr. 5, S.22-29; Aus: http://delivery.acm.org/10.1145/1020000/1015541/p22-rosenberg.pdf?key1=1015541&key2=7498203011&coll=GUIDE&dl=GUIDE&CFID=33833163&CFTOKEN=83301848 am 29.09.2005.

[Schliwka 1998]
Schliwka, Anja (1998): Teamorientiertes, medienübergreifendes Know-how-Management in verteilten Umgebungen. Shaker Verlag, Aachen 1998.

[Sewell 2005]
Sewell, Katherine (2005): Introduction to the IBM Workplace Managed Client Developer Toolkit. IBM, 2005; Aus: http://www-128.ibm.com/developerworks/lotus/library/wmc-toolkit/ am 27.09.2005.

[Spolsky 2001]
Spolsky, Joel (2001): User Interface Design for Programmers. Springer, Heidelberg 2001.

[Stelzer 2002]
Stelzer, Dirk (2002): Warum ist Software so fehlerhaft - und was kann man dagegen tun?. Freiberg 2002; Aus: http://www.wiwi.tu-freiberg.de/wi/courses/guest_lectures/2002/warum_ist_software_so_fehlerhaft.pdf am 29.09.2005.

[Stimmler 2002]
Stimmler, Uta (2002): Ansatzpunkte für ein Controlling lern- und wissensintensiver Bereiche von Industrieunternehmen. Brandenburgische technische Universität(Doktorarbeit), Potsdam 2002.

[Sulzmaier 2002]
Sulzmaier, Sonja (2002): E-Usability. In:Usability – nutzerfreundliches Web-Design. Hrsg.: Beier, Markus; von Gizycki, Victoria. Springer, Heidelberg 2002.

[Voltz 1990]
Voltz, Hannspeter (1990): Das große Wörterbuch der Computer-Fachbegriffe. Signum-Medien-Verlag, München 1990.

[Wild, Herges 2000]
Wild, Martin ; Herges, Sascha (2000): Total Cost of Ownership (TCO) - Ein Überblick. In: Arbeitspapiere WI, Nr. 1/2000, Hrsg.: Lehrstuhl für Allg. BWL und Wirtschaftsinformatik, Johannes Gutenberg Universität, Mainz 2000.

[Zänger 2005]
Zänger, Roland (2005): Konzeption eines generischen Vorgehensmodells zur Entwicklung kollaborativer Applikationen und prototypische Implementierung am Beispiel einer J2EE-Plattform. Universität Paderborn (Diplomarbeit), Paderborn 2005; Aus: http://gcc.uni-paderborn.de/www/WI/WI2/wi2_lit.nsf/1194fe21b0962131c1256f3c003fe9fc/299ade1fec78f1c5c1256cfd00541e6f/$FILE/Diplomarbeit_Roland_Zaenger.pdf am 29.09.2005.

8 Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

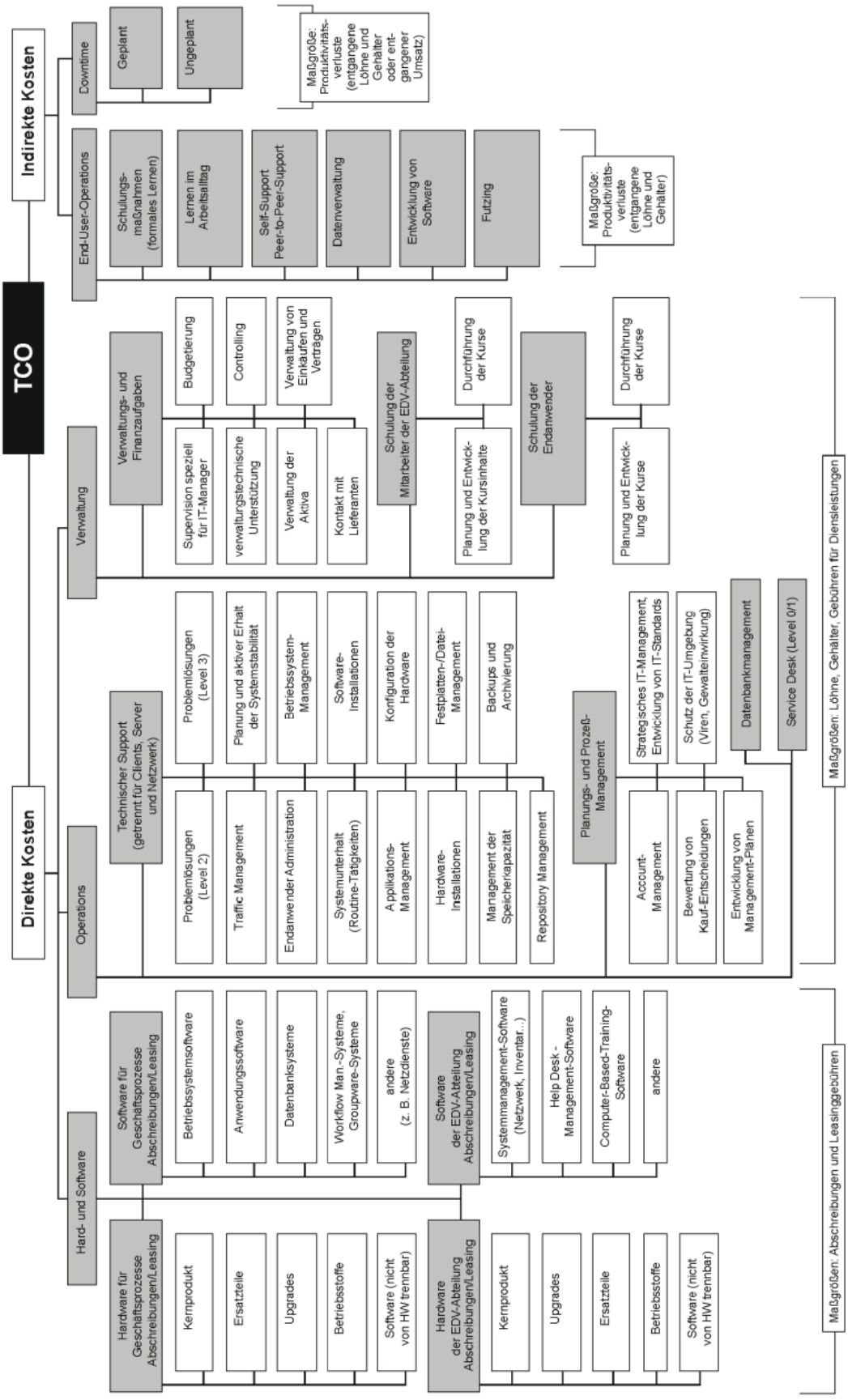
Paderborn, den ....

(Datum) (Unterschrift)

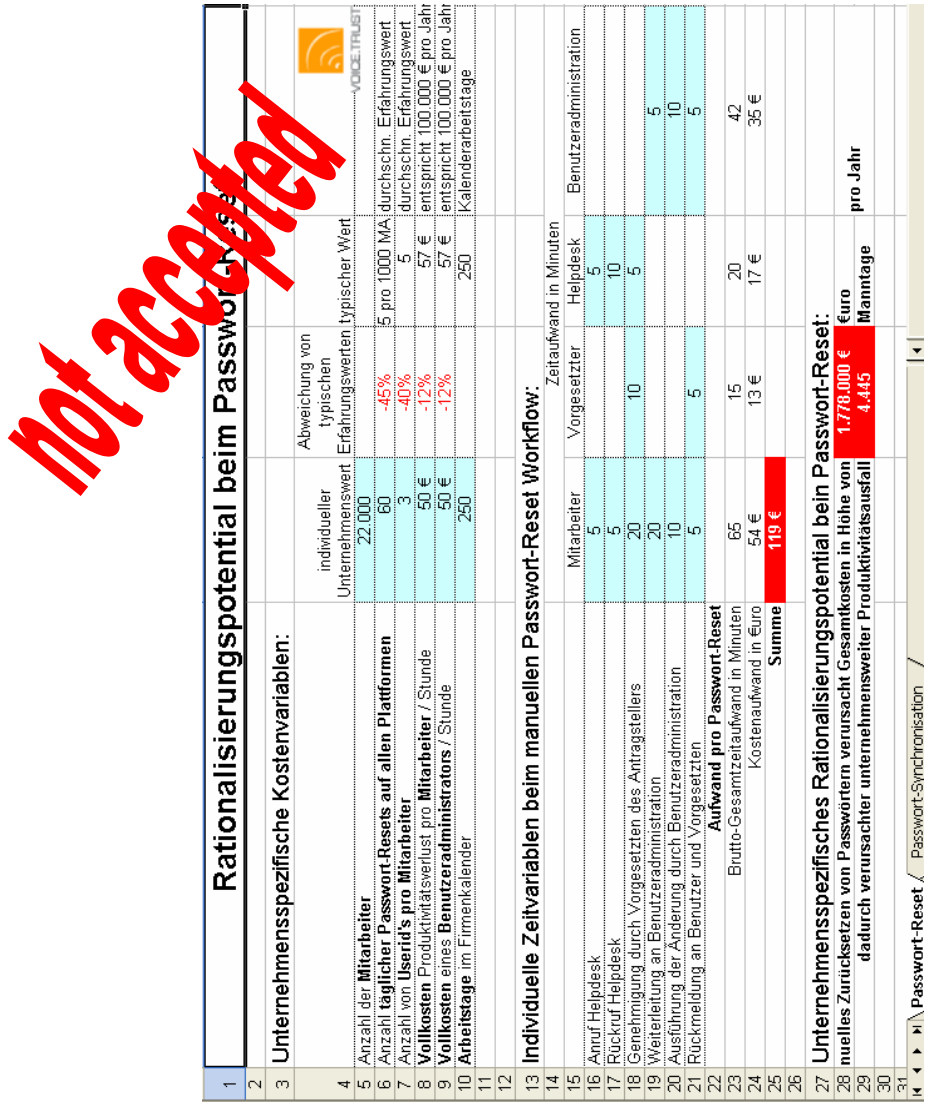
9 Anhang

a. Beispiel für ein TCO-Modell der Gartner Group
b. Beispiel für das Problem der Glaubwürdigkeit von Usability Berechnungen
c. WMC Komponente K-Pool – Benutzerrollen und deren Rechte
d. WMC Komponente K-Pool – Aktivitätendiagramm Login
e. WMC Komponente K-Pool – Datenbankschema
f. WMC Komponente K-Pool – Klassendiagramme
g. WMC Komponente K-Pool - Entwicklungspotentiale
h. Informationen zur beiligenden CD

a. Beispiel für ein TCO-Modell der Gartner Group



b. Beispiel für das Problem der Glaubwürdigkeit von Usability Berechnungen



Rationalisierungspotential beim Passwort-Reset

not accepted

Unternehmensspezifische Kostenvariablen:

	individueller Unternehmenswert	Abweichung von typischen Erfahrungswerten	typischer Wert	
Anzahl der **Mitarbeiter**	22.000			
Anzahl **täglicher Passwort-Resets auf allen Plattformen**	60	-45%	5 pro 1000 MA	durchschn. Erfahrungswert
Anzahl von **Userid's pro Mitarbeiter**	3	-40%	5	durchschn. Erfahrungswert
Vollkosten Produktivitätsverlust pro **Mitarbeiter** / Stunde	50 €	-12%	57 €	entspricht 100.000 € pro Jahr
Vollkosten eines **Benutzeradministrators** / Stunde	50 €	-12%	57 €	entspricht 100.000 € pro Jahr
Arbeitstage im Firmenkalender	250		250	Kalenderarbeitstage

Individuelle Zeitvariablen beim manuellen Passwort-Reset Workflow:

	Zeitaufwand in Minuten			
	Mitarbeiter	Vorgesetzter	Helpdesk	Benutzeradministration
Anruf Helpdesk	5		5	
Rückruf Helpdesk	5		10	
Genehmigung durch Vorgesetzten des Antragstellers	20	10	5	
Weiterleitung an Benutzeradministration	20			5
Ausführung der Änderung durch Benutzeradministration	10			10
Rückmeldung an Benutzer und Vorgesetzten	5	5		5
Aufwand pro Passwort-Reset				
Brutto-Gesamtzeitaufwand in Minuten	65	15	20	42
Kostenaufwand in €uro	54 €	13 €	17 €	35 €
Summe	**119 €**			

Unternehmensspezifisches Rationalisierungspotential bein Passwort-Reset:

nuelles Zurücksetzen von Passwörtern verursacht Gesamtkosten in Höhe von	**1.778.000 €**	**€uro**
dadurch verursachter unternehmensweiter Produktivitätsausfall	**4.445**	**Manntage**
		pro Jahr

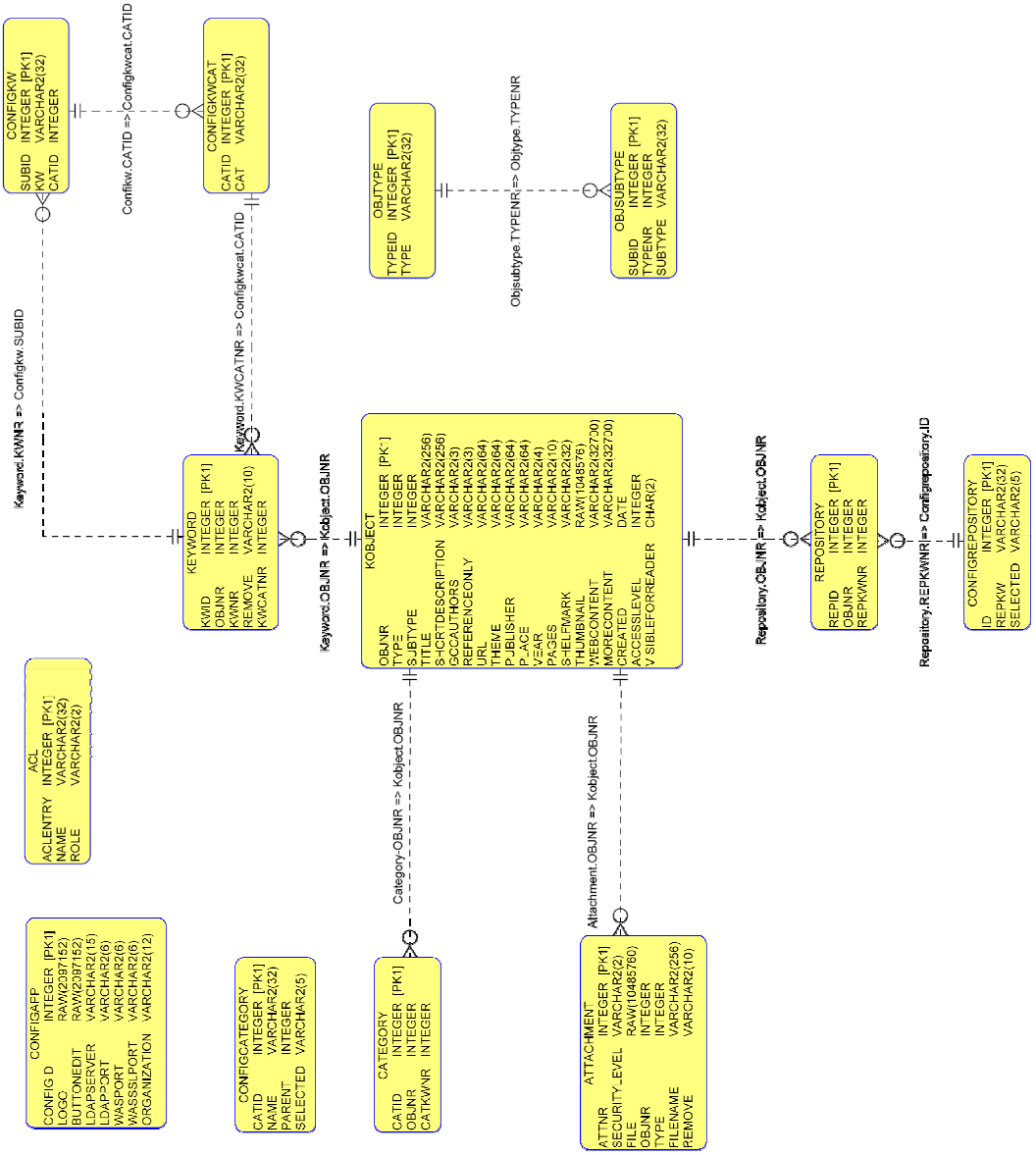
Passwort-Reset | Passwort-Synchronisation

c. WMC Komponente K-Pool – Benutzerrollen und deren Rechte

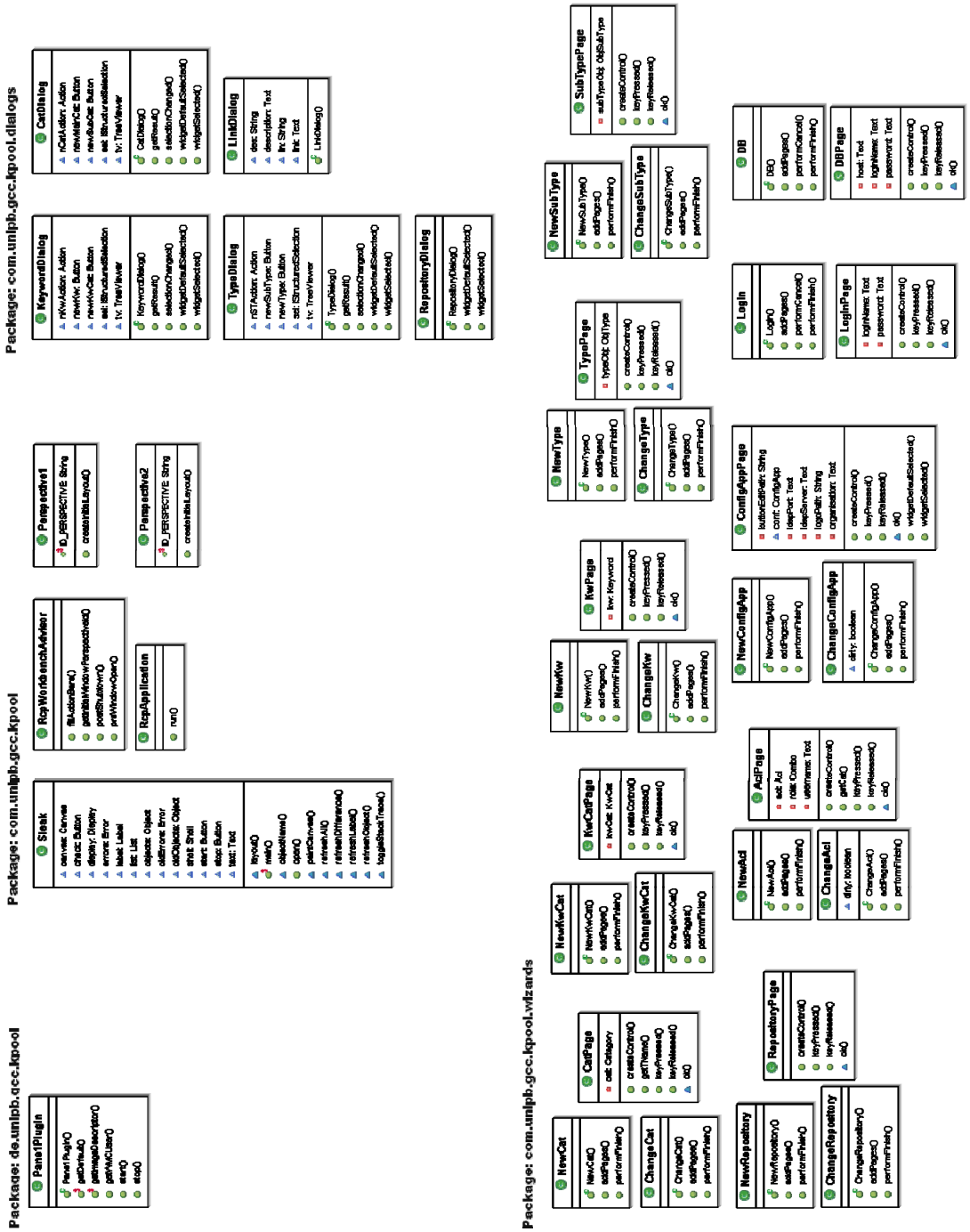
Name der Rolle	Rechte der Rolle
reader	• entspricht der Rolle des anonymen Benutzers • kein Zugriff auf „morecontent“ • lesender Zugriff auf für reader: – freigegebene K-Objects – freigegebene Dateianhänge
private reader	• kein Zugriff auf „morecontent“ • lesender Zugriff auf für private reader: – freigegebene K-Objects – freigegebene Dateianhänge
editor	• kann neue K-Objects und Ordnungskategorien erstellen • schreibender und lesender Zugriff auf Ordnungskategorien • schreibender und lesender Zugriff auf für editor: – freigegebene K-Objects – freigegebene Dateianhänge
private editor	• kann neue K-Objects und Ordnungskategorien erstellen • schreibender und lesender Zugriff auf Ordnungskategorien • kann bestehende K-Objects löschen • schreibender und lesender Zugriff auf für private editor: – freigegebene K-Objects – freigegebene Dateianhänge
admin	• Inhaltlich: Rechte eines private editor • Administration: – Benutzerverwaltung: anlegen / ändern / löschen von Benutzern – Konfiguration: anlegen / ändern / löschen von Anwendungsprofilen

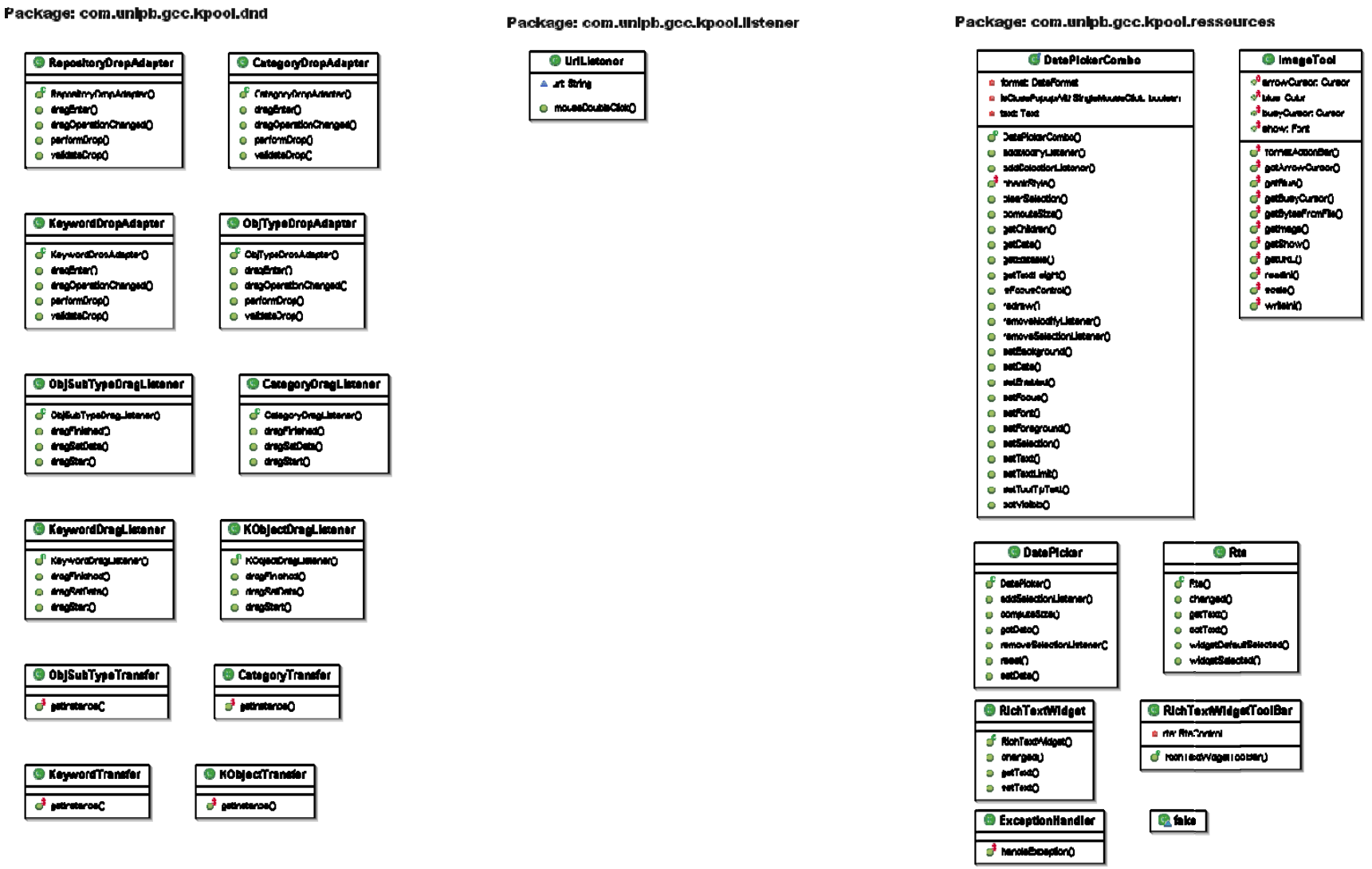
d. WMC Komponente K-Pool – Aktivitätendiagramm Login

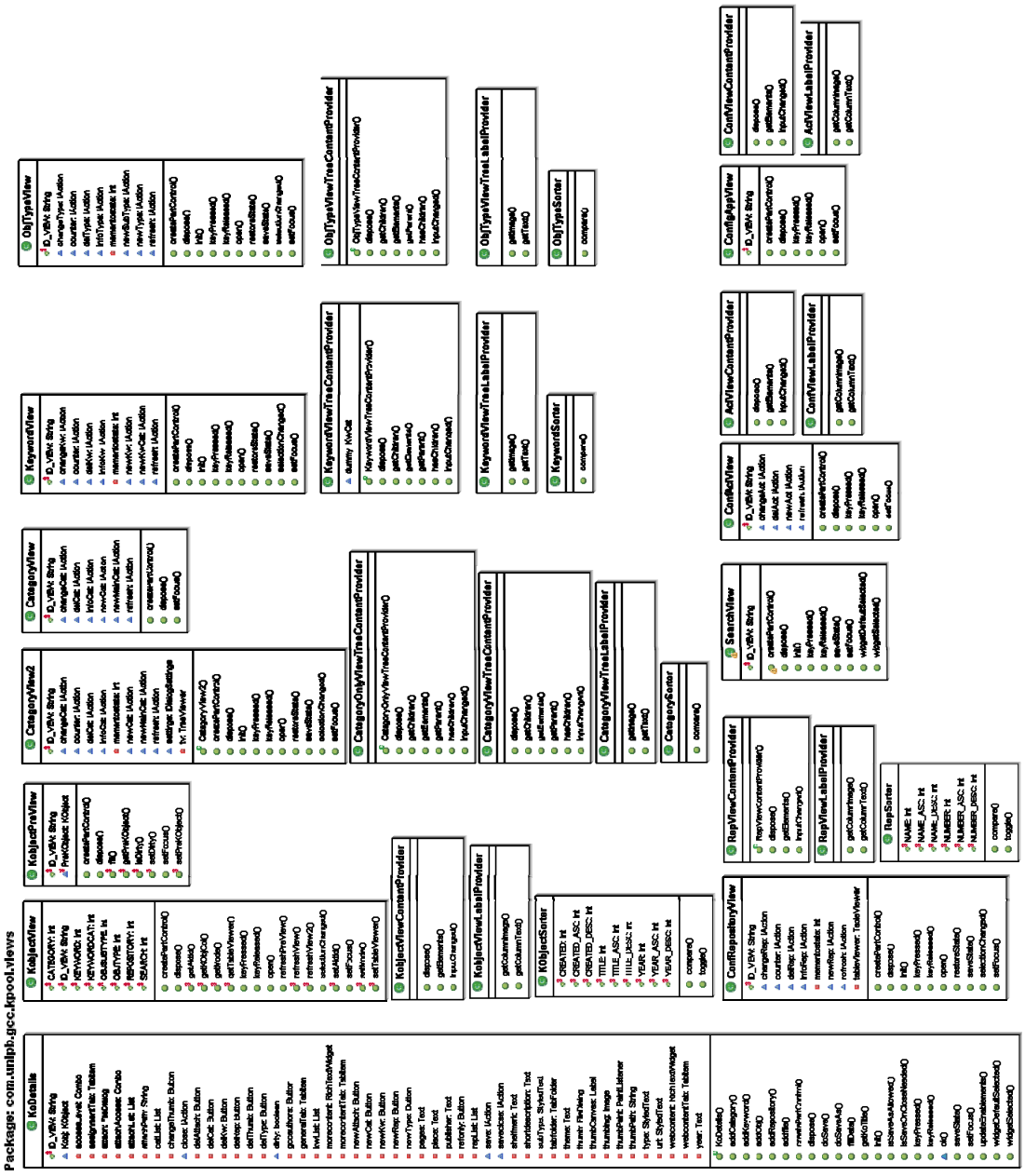
e. WMC Komponenten K-Pool – Datenbankschema

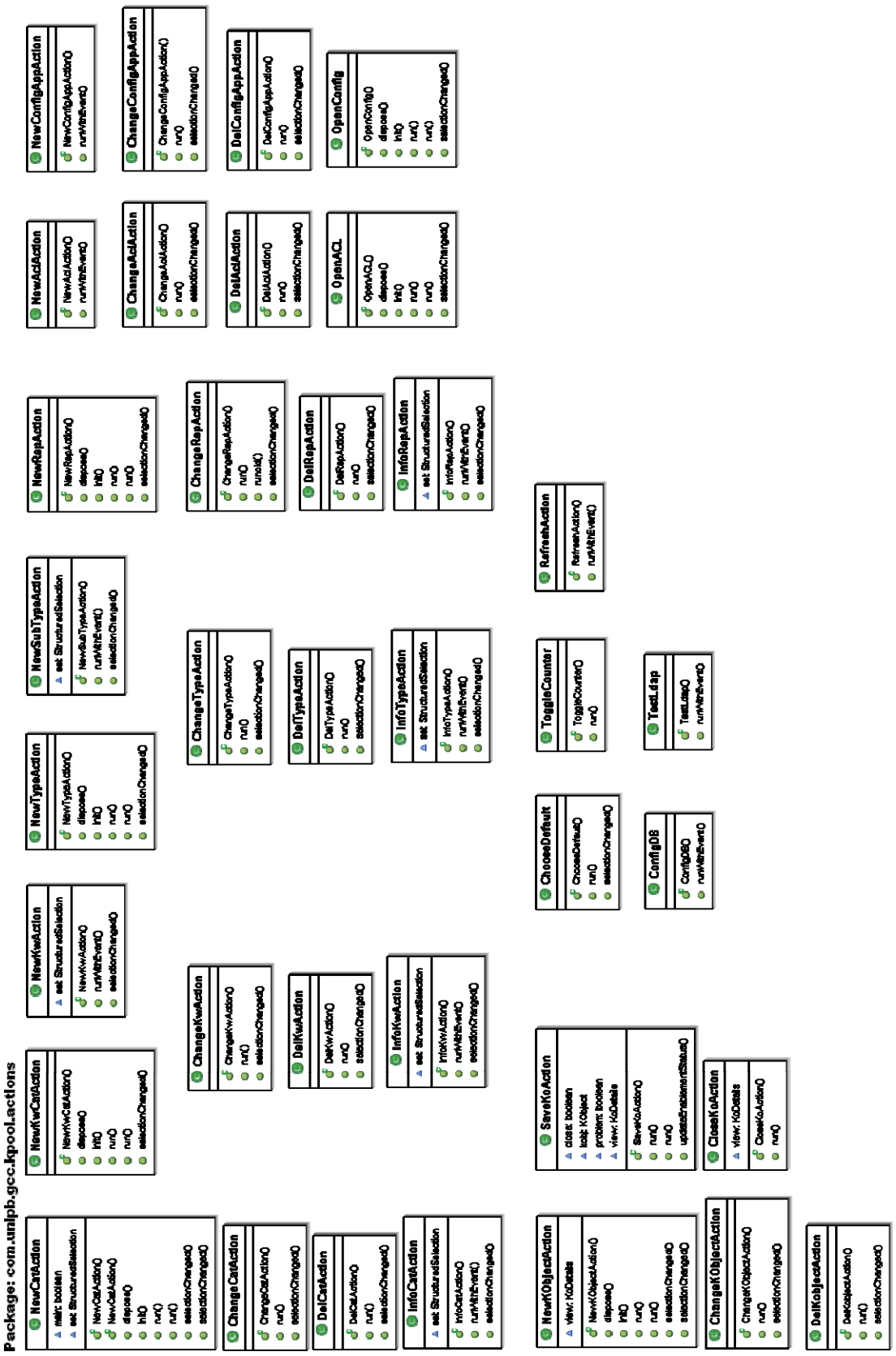


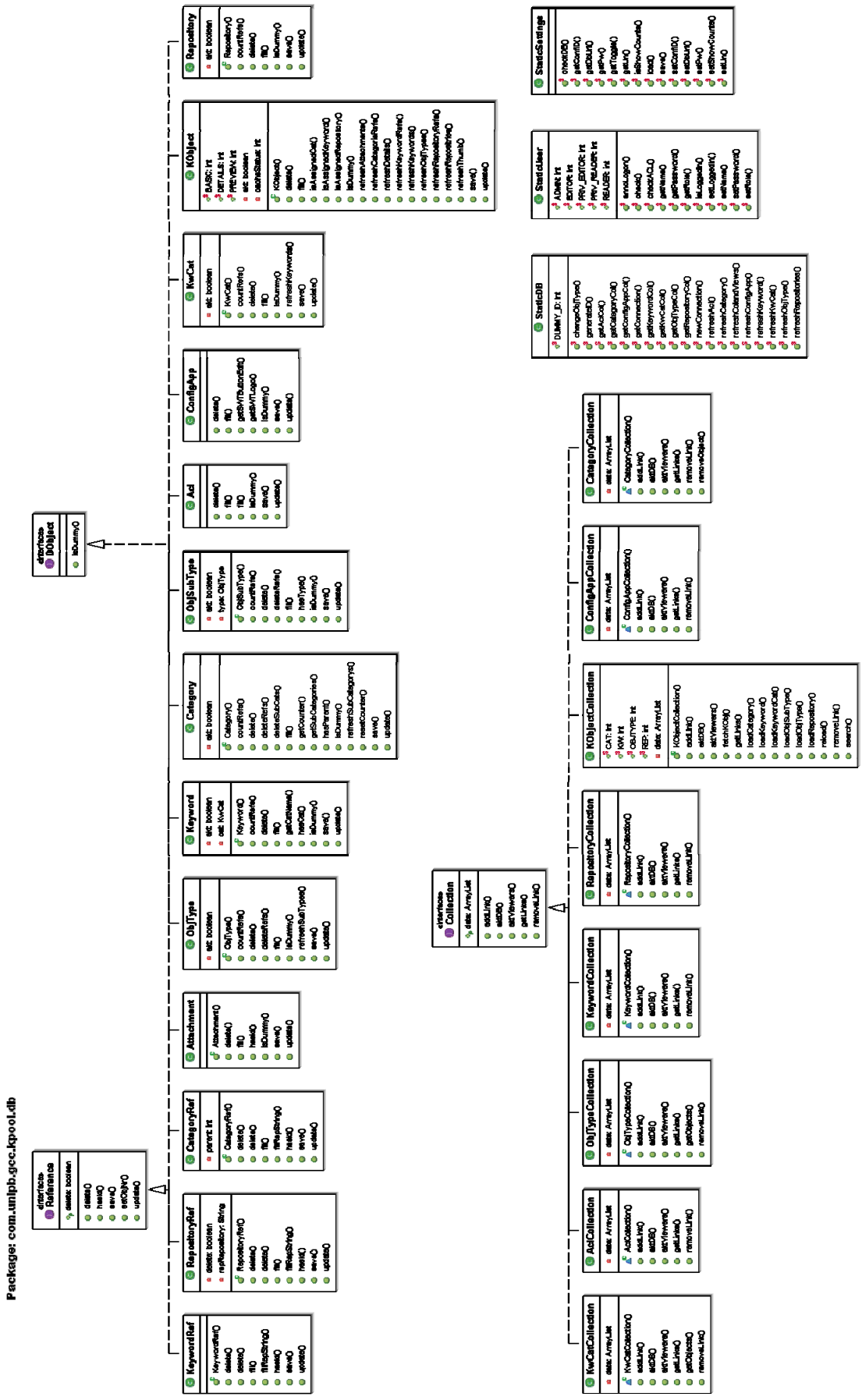
f. WMC Komponente K-Pool – Klassendiagramme




Package: com.unipb.gcc.kpool.dnd
RepositoryDropAdapter
dragEnter()
dragOperationChanged()
performDrop()
validateDrop()
CategoryDropAdapter
dragEnter()
dragOperationChanged()
performDrop()
validateDrop()
KeywordDropAdapter
dragEnter()
dragOperationChanged()
performDrop()
validateDrop()
ObjTypeDropAdapter
dragEnter()
dragOperationChanged()
performDrop()
validateDrop()
ObjSubTypeDragListener
dragFinished()
dragSetData()
dragStart()
CategoryDragListener
dragFinished()
dragSetData()
dragStart()
KeywordDragListener
dragFinished()
dragSetData()
dragStart()
KObjectDragListener
dragFinished()
dragSetData()
dragStart()
ObjSubTypeTransfer
getInstance()
CategoryTransfer
getInstance()
KeywordTransfer
getInstance()
KObjectTransfer
getInstance()
Package: com.unipb.gcc.kpool.listener
UrlListener
mouseDoubleClick()
Package: com.unipb.gcc.kpool.ressources
DatePickerCombo
format: DateFormat
text: Text
addSelectionListener()
clearSelection()
getChildren()
getData()
getText()
setFocus()
setFont()
setForeground()
setSelection()
setText()
setTextLimit()
setVisible()
ImageTool
arrowCursor: Cursor
busyCursor: Cursor
show: Font
getArrowCursor()
getBusyCursor()
getImage()
getShow()
scale()
DatePicker
DatePicker()
addSelectionListener()
getData()
reset()
setData()
Rte
Rte()
changed()
getText()
setText()
widgetDefaultSelected()
widgetSelected()
RichTextWidget
RichTextWidget()
getText()
setText()
RichTextWidgetToolBar
ExceptionHandler
handleException()
fake


Package: com.uniph.gcc.kpool.views
KoDetails
KoObjectView
KoSelectView
KoSelectPreView
CategoryView
CategoryView2
KeywordView
ObjTypeView
ConfigAppView
ConfRecView
SearchView
ConfRepositoryView
RepViewContentProvider
RepViewLabelProvider
KoSelectViewContentProvider
KoSelectViewLabelProvider
KoSelectSorter
CategoryOnlyViewTreeContentProvider
CategoryViewTreeContentProvider
CategoryViewTreeLabelProvider
CategorySorter
KeywordViewTreeContentProvider
KeywordViewTreeLabelProvider
KeywordSorter
ObjTypeViewTreeContentProvider
ObjTypeViewTreeLabelProvider
ObjTypeSorter
AdvViewContentProvider
AdvViewLabelProvider
ConfViewContentProvider
ConfViewLabelProvider


Package: com.unipb.gcc.kpool.actions
NewCatAction
NewKwCatAction
NewKwAction
NewTypeAction
NewSubTypeAction
NewRepAction
NewActAction
NewConfigAppAction
ChangeCatAction
ChangeKwAction
ChangeTypeAction
ChangeRepAction
ChangeActAction
ChangeConfigAppAction
DelCatAction
DelKwAction
DelTypeAction
DelRepAction
DelActAction
DelConfigAppAction
InfoCatAction
InfoKwAction
InfoTypeAction
InfoRepAction
OpenACL
OpenConfig
NewKoObjectAction
SaveKoAction
ChooseDefault
ToggleCounter
RefreshAction
ChangeKoObjectAction
CloseKoAction
Config08
TestLdap
DelKoObjectAction


Package: com.uniph.gcc.kpool.db
interface Referenca
interface DObject
Repository
KwCat
KObject
ConfigApp
Acl
ObjSubType
Category
Keyword
ObjType
Attachment
CategoryRef
RepositoryRef
KeywordRef
StaticSettings
StaticUser
StaticDB
interface Collection
CategoryCollection
ConfigAppCollection
KObjectCollection
RepositoryCollection
KeywordCollection
ObjTypeCollection
AclCollection
KwCatCollection

g. WMC Komponente K-Pool – Entwicklungspotentiale

Problem-kategorie	Bestehendes Problem / Einschränkung	Ursache des Problems / mögliche Erweiterung	Mögliche Lösungsansätze
Allgemein	Quellcode der Komponente ist teilweise nicht optimal strukturiert.	Im Rahmen der prototypischen Implementierung mit dieser neuen Technologie wurden oft mehrer Wege zur Zielerreichung geteste, worunter die Strukturierung gelitten hat	Refactoring Projekt für den vorhandenen Quellcode durchführen
	Inkompatibilität zum K-Pool Everyplace von Roland Zänger	Da es eine Projektanforderung war Richt Text Editierfunktionen im Client anzubieten, werden die "Webcontent" und "morecontent" Felder als HTML gespeichert. Da dies im K-Pool Everyplace nicht so implementiert ist wird dieser die Tags in seiner Ansicht mit anzeigen	Erweiterung des K-Pool Everyplace um eine Richt Text Editierfunktion durch Nutzung einer entsprechenden JSF Komponente
	Internationalisierung	Als Prototyp wurden die erforderlichen Benutzerausgaben zunächst fest in den Quelltext geschrieben.	Die im Quelltext enthaltenen Strings können mit entsprechenden Tools aus dem Quelltext in eine entsprechende Ressource exportiert werden und dann einfach neu Sprachen hinzugefügt werden. So das sich die Komponente automatisch auf die im Workplace benutzte Sprache einstellen kann
	Verwaltung der Benutzer rudimentär	Der Prototyp verfügt nur über eine sehr einfache Benutzungsoberfläche zur Verwaltung der Benutzer. So muss der entsprechende Benutzername von Hand eiingegeben werden	Erweiterung der Benutzerverwaltung mit einem Dialog der die Auswahl von Benutzernamen aus dem Workplace Verzeichnis oder alternativ dem eingestellten LDAP Server ermöglicht
Suche	Die Suche innerhalb der K-Objects ist für das "webcontent"und "morecontent" Feld bedingt abhängig von der Groß und Kleinschreibung der Inhalte bzw. des Suchbegriffs	Die für diese beiden Datenfelder verwendeten großen Textdatentypen (LongVarChar) lassen eine Benutzung des SQL Befehls "UPPER" nicht zu. Zur Zeit wird die Anfrage so modifiziert das nach 4 unterschiedlichen Varianten des Begriffs gesucht wird. ("SUchanfrage" wird auch zu "suchanfrage" / "sUchanfrage" / "SUCHANFRAGE") Dies sollte viele (aber nicht alle) praktische Fälle abfangen.	Prüfen ob diese Einschränkung der Datenbank umgangen werden kann oder weitere Modifikation der Suchanfrage so das alle Fälle abgedeckt werden. (Ist durch die resultierende Länge der Suchanfrage allerdings eher bei Umsetzung einer 3-Tier Architektur zu empfehlen)

Problem-kategorie	Bestehendes Problem / Einschränkung	Ursache des Problems / mögliche Erweiterung	Mögliche Lösungsansätze
	Die Suche liefert unter bestimmten Umständen zu viele Ergebnisse zurück	Da im "webcontent" und "morecontent" Feld auch Formatierungsinformationen in Form von HTML gespeichert sind werden Suchanfragen nach HTML Schlüsselwörtern stets alle in denen das entsprechende Feld gefüllt ist zurückliefern	Filtern der Suchwörter und entsprechende Einschränkung der Suchanfrage auf Felder die kein HTML enthalten auf dem Client oder die Filterung der Ergebnisse. Doppelte Speicherung der entsprechenden Felder einmal mit und einmal ohne HTML (nur zur Suche).
	In der vorliegenden Version ist keine Suche in Dateinahängen möglich	Aufgrund der 2-Tier Architektur kann, da sonst das Datenaufkommen zwischen Client und DB-Server zu groß wird, ene Suche nur mit SQL Befehlen implementiert werden. SQL bietet allerdings keine Funktionen zur Suche in Binärfeldern (BLOB).	In Kombination mit einer 3-Tier Architektur kann eine entsprechende Suche, möglicherweise unter Nutzung vorhandener Lösungen, für die Anwendung implemntiert werden
Datenbankdesign / Datenanbindung	Aktualisierung der Ansichten bei veränderten Daten findet nicht unaufgefordert statt	Durch die 2-Tier Architektur, und die damit auf SQL festgelegte Kommunikation, kann der Datenbankserver keine Änderungsbenachrichtigungen an die Clients schicken. Auf eine vom Client getriebene Polling Architektur wurde aus Performancegründen verzichtet	Ein Server, bei dem die Clients angemeldet sind, kann in einer beschriebenen 3 Tier Architektur entsprechende Benachrichtigungen an die Clients verschicken. Alternativ: Aufnahme des Zeitpunkts der letzten Änderung in das Datenmodell und über diese Angabe ein relativ effizientes Polling realisieren
	Datenmodell sichert nicht schon durch das Design die Konsistenz	Einige implizite Fremdschlüsselbeziehungen sollten explizit im Datenmodell definiert werden während eine Beziehung überflüssig ist. Änderungen wurden hier nicht vorgenommen, da diese auch parallel im K-Pool Everyplace erfolgen sollten.	Hinzufügen der FK-Beziehungen: CONFIGCATEGORY: PARENT=> CATID CATEGORY.CATID => CONFIGCATEGORY.CATID KOBJECT.TYPE => OBJTYPE.TYPEID KOBJECT.SUBTYPE => OBJTYPE.SUBID Entfernen der Beziehung ist über 2 bestehende definiert: KEYWORD.KWCATNR => CONFIGKWCAT.CATID und entsprechende Tests/ Anpassungen im K-Pool Everyplace und in der K-Pool WMC Komponenten durchführen
	Im Datenmodell werden unterschiedliche Bezeichnungen für semantisch gleiche Funktionen benutzt	Booleanvariablen werden im Datenmodell zum Teil als "YES/NO" an anderen Stellen aber wiederum als "0/1" gespeichert	Datenmodell vereinheitlichen und entsprechende Tests/ Anpassungen im K-Pool Everyplace und in der K-Pool WMC Komponenten durchführen

Problem-kategorie	Bestehendes Problem / Einschränkung	Ursache des Problems / mögliche Erweiterung	Mögliche Lösungsansätze
Integration in den WMC	Offlinefähigkeit ist nicht vorhanden	Da die Kommunikation zwischen Datenbank und Client direkt erfolgt, müsste eine Offlinefähigkeit von Grund auf selbst implementiert werden. Auf der anderen Seite bietet der WMC Client (noch) keine einfach zu nutzende generische Schnittstelle für die Replikation von beliebigen relationalen Daten	Entwicklung des im WMC verwendeten SyncML Standards abwarten und bei entsprechender Verfügbarkeit die Datenhaltung der Komponente auf die Nutzung der Workplace Datenbankdienst umstellen. Prüfen ob zusätzlich eine Serverkomponente (==> 3-Tier Architektur) erstellt werden kann um die vor allem bei der Suche auftauchenden Probleme zu lösen.
	Integration in weitere Workplace Dienste	Neben den bereits genutzten Workplace Ressourcen (z.B. Browser und Updatemechanismus) können weitere Workplace Dienst in die Komponente eingebunden werden	In Kombination mit einer Erweiterung des Datenmodells (z.B. Speicherung des Workplace Benutzernamens des Autors) können (Instant) Messaging Funktionalitäten genutzt werden

h. Informationen zur beiliegenden CD

Inhalt der CD:

- Onlineversion dieser Arbeit im Microsoft Word und Acrobat Reader Format
- Daten zur Veröffentlichung im K-Pool (Foto, Kerngrafiken, Abstract (ENG/DE)
- Kopien der zitierten Online verfügbaren Quellen
- Sourcecode der WMC Komponente K-Pool in Form eines Eclipse Workspaces
- Durch das WMC Toolkit erstellte Dateien mit deren Hilfe sich die Komponente auf einem Workplace Server installieren lässt
- DDL-Datei mit der eine entsprechende DB2 Datenbank erstellt werden kann
- Informationen zu Installation und Konfiguration der WMC-Komponente K-Pool