GMD Report 141

GMD –
Forschungszentrum
Informationstechnik
GmbH

Anita Isabella Pelka

Integration von Software- und Usability-Engineering am Beispiel des V-Modells

Juli 2001

© GMD 2001

GMD –
Forschungszentrum Informationstechnik GmbH
Schloß Birlinghoven
D-53754 Sankt Augustin
Germany
Telefon +49 -2241 -14 -0
Telefax +49 -2241 -14 -2618
http://www.gmd.de

In der Reihe GMD Report werden Forschungs- und Entwicklungsergebnisse aus der GMD zum wissenschaftlichen, nichtkommerziellen Gebrauch veröffentlicht. Jegliche Inhaltsänderung des Dokuments sowie die entgeltliche Weitergabe sind verboten.

The purpose of the GMD Report is the dissemination of research work for scientific non-commercial use. The commercial distribution of this document is prohibited, as is any modification of its content.

Die vorliegende Veröffentlichung entstand im/
The present publication was prepared within:
Institut für Autonome intelligente Systeme (AiS)
Institute for Autonomous intelligent Systems (AiS)

Anschrift der Verfasserin/Address of the author:
Anita Isabella Pelka
Postfach 7463
D-53074 Bonn

ISSN 1435-2702

Kurzfassung

Die vorliegende Diplomarbeit beschreibt Aspekte der methodischen Vorgehensweise bei der Integration von Usability-Engineering und Software-Engineering. Anhand eines Prozess-Modells für das Usability-Engineering (UE) wird gezeigt, was für die benutzerorientierte Gestaltung von Software analytisch und konstruktiv zu tun ist, und zwar hauptsächlich bei der Vorbereitung von Software-Entwicklungsprojekten.

Die Beschreibung des Prozess-Modells für das Usability-Engineering erfolgt am Beispiel des V-Modells, indem das V-Modell um Aspekte des UE erweitert und modifiziert wird. Hierbei werden die einschlägigen Normen berücksichtigt, vor allem ISO 13407, ISO 12207 und ISO 9241.

Bei der praktischen Umsetzung (Evaluierung) des UE-Prozess-Modells wurde die Erfahrung gemacht, dass diese Vorgehensweise nicht nur die Nutzungsqualität des zu entwickelnden Produkts sichern hilft, sondern auch dazu beiträgt, dass Entwicklungprojekte wegen ihrer besseren Vorbereitung im Kosten- und Zeitrahmen bleiben.

Künftig wird Nutzungsqualität von Softwareprodukten ein entscheidender Wettbewerbs-faktor werden, denn es wird erfahrenen Anwendern und Benutzern nicht mehr ausreichen, dass Software nur zuverlässig funktioniert. Der Prozess der Software-Entwicklung sollte auf diese Herausforderung vorbereitet sein.

Schlagwörter: Nutzungsqualität, Gebrauchstauglichkeit, Usability, Usability-Engineering, Prozessmodell, V-Modell, Design-Use-Cycle, DATech, User-Centred-Design, ISO Normen

Abstract

This diploma thesis deals with aspects of a methodical approach to integrate usability engineering and software engineering. Based on a process model of usability engineering it is described what needs to be done in a user-centred process of software development, particularly during the analytical and constructive preparation of development projects.

The process model of usability engineering is described following the V-model, which is a recognized software-engineering process model in Germany. The V-model has been extended and modified by the usability process model. Pertinent international standards have been considered, such as ISO 13407, ISO 12207, and ISO 9241.

In practicing the usability-engineering process model it turned out that this approach is beneficial for ensuring the product's usability. Moreover, development projects could be accomplished within time and budget constraints, due to the better preparation of the projects.

Usability will become a key competitive factor in the software market, since the more mature customers and users will no longer be satisfied with a product that merely works correctly and reliably. The software development process should be well prepared for that challenge.

Keywords: usability, usability engineering, process model, V-model, design-use cycle, DATech, user-centred design, standards

„Wir tätigen etwas heute,
um es morgen noch besser zu tun“.

Vorwort der Autorin

Die Idee der vorliegenden Arbeit entstand im Forschungsprojekt „Usability-Engineering“ des Institutes für Autonome intelligente Systeme (AiS) des GMD-Forschungszentrums Informationstechnik.

An dieser Stelle möchte ich mich bei Herrn Prof. Dr. Armin B. Cremers für die Bereitschaft bedanken, diese Arbeit zu betreuen. Ich danke auch Herrn PD Dr. Joachim Hertzberg und Herrn Dr. Volker Wulf für ihre Unterstützung bei der Diplomarbeit.

Dr. Wolfgang Dzida im Institut AiS der GMD verdanke ich die Möglichkeit, mich mit dem zukunftsweisenden Thema „Usability-Engineering“ auseinander zu setzen. Er hat mich fachlich sehr gut betreut und nahm sich immer Zeit, meine vielfältigen Fragen zu beantworten. Ich möchte mich an dieser Stelle für seine stete Bereitschaft, mich zu unterstützen, bedanken. Es hat mir grossen Spass gemacht, mit ihm und seinen Kollegen, vor allem Frau Dipl.-Inform. Regine Freitag, fachliche Diskussionen zu führen. Ihre konstruktiven Kritiken, die ich immer gern angenommen habe, trugen zum Inhalt der vorliegenden Diplomarbeit bei.

Weiterhin möchte ich mich bei der „DATech Arbeitsgruppe Usability-Engineering“ bedanken, vor allem bei Herrn Dipl.-Ing. Thomas Geis, der mir ermöglicht hat, an den Arbeitssitzungen teilzunehmen, um aus ihren Erfahrungen zu lernen.

Bonn, 15. Dezember 2000

Anita Isabella Pelka

Hinweis:
In dieser Arbeit werden die Regeln der neuen Rechtschreibung angewandt. Ausserdem habe ich den Buchstaben ß immer durch ss ersetzt.

Vorwort des Betreuers

Wer glaubt, gebrauchstaugliche Software entwickeln zu können, indem man sich streng und allein an die Vorgaben des V-Modells hält, der unterliegt etwa dem gleichen Irrtum wie wenn man glaubt, man könne Anwendungsprogramme entwickeln, nur weil man gelernt hat zu programmieren. Das V-Modell ist eine notwendige, aber keine hinreichende Voraussetzung zur Entwicklung gebrauchstauglicher Software. Dies ist eine mögliche Schlussfolgerung aus der vorliegenden Diplomarbeit.

Die Diplomarbeit ist unter anderem am AiS Institut für Autonome intelligente Systeme der GMD betreut worden. Die Arbeit versucht eine Brücke zwischen zwei unabhängig von einander entstandenen Teildisziplinen zu schlagen: Software-Engineering und Usability-Engineering. Der Versuch ist insofern gelungen als gezeigt wird, dass ein existierendes Vorgehensmodell um ein Planungsprojekt ergänzt werden kann, das im Formalismus des V-Modells beschrieben wird. Konstruktive Qualitätssicherung beginnt bereits vor Beginn eines Software-Entwicklungsprojekts, und zwar mit der Entwicklung von Anforderungen an die Gebrauchstauglichkeit (Usability) und deren Validierung im Konsens zwischen Anwender, Benutzern und Hersteller. Die Autorin zeigt, welche Übergänge es zwischen dem Planungsprozess und dem Vorgehensmodell gibt. Somit wird die Hoffnung genährt, das die gezeigten Übergänge künftig besser genutzt werden in der Kooperation zweier Teildisziplinen.

Berlin, den 10. August 2001

Dr. Wolfgang Dzida

Inhaltsverzeichnis

1 EINFÜHRUNG 9

1.1 EINLEITUNG 10

1.1.1 Ziel der Diplomarbeit 10

1.1.2 Ein Prozess-Modell für Usability-Engineering 11

1.1.3 Erweiterungen des V-Modells um Usability-Engineering 12

1.2 PROBLEMDARSTELLUNG 12

1.2.1 Das Synonymen-Problem 13

1.2.2 Probleme bei Vernachlässigung der Nutzungs-Qualität von Software 14

1.2.3 Die Kosten und das Problem mit der Produktivität 15

1.2.4 Die Bildschirmarbeitsverordnung und die ISO-Normen 16

2 DAS USABILITY-ENGINEERING 21

2.1 VEREINBARUNG DER TERMINOLOGIENUTZUNG 21

2.2 ASPEKTE DES USABILITY-ENGINEERING 23

2.2.1 Aufgaben des Usability-Engineering 25

2.2.2 Das Nutzungskonzept 25

2.2.3 Erhebung von Szenarien 27

2.2.4 Die Entwicklung der Anforderungs-Definition 36

2.2.5 „Discount Usability“ 46

2.3 DIE NORM ISO 13407 49

3 DAS V-MODEL 53

3.1 ÜBERBLICK ÜBER DAS V-MODELL 53

3.1.1 Definitionen, Strukturen und das organisatorische Umfeld 57

3.1.2 Die Aktivitäten des Submodells SE 62

3.2 SOFTWARE-LIFECYCLE-PROCESSES IN ISO 12207 64

3.2.1 Überblick über die ISO 12207 64

3.2.2 Abdeckung der Prozesse der ISO 12207 durch das V-Modell 66
3.3 EIGENSCHAFTEN DES V-MODELLS 67
3.4 FAZIT 68
4 EIN PROZESS-MODELL FÜR USABILITY-ENGINEERING 71
4.1 DER DIALEKTISCHE PROZESS 74
4.2 DAS UE-PROZESS-MODELL 76
4.2.1 Die Usability-Analyse Aktivität 78
4.2.2 Die Anforderungs-Entwicklung Aktivität 84
4.2.3 Die System-Machbarkeitsanalyse Aktivität 90
4.2.4 Die SW-Pflege Aktivität 94
4.3 DIE ZERTIFIZIERUNG, ISO-NORMEN UND DAS UE-PROZESS-MODELL 98
5 EVALUIERUNG 103
5.1 ERHEBUNG VON KONTEXT-SZENARIEN 103
5.2 ENTSTEHUNG DER SYSTEM-IDEE 112
5.3 ABBAU DER DIALEKTISCHEN BARRIERE 113
5.4 ERSTELLUNG EINES USE-SZENARIOS 113
5.5 SYSTEM-MACHBARKEITSENTWURF 117
5.6 ABBAU DER SYNTHESEBARRIERE 117
5.7 PFLEGE-PROZESS 118
6 AUSBLICK 119
7 GLOSSAR 121
8 LITERATURVERZEICHNIS 127

1 Einführung

Die Produktion und der Einsatz moderner Informationstechnologie ist für die Wirtschaft von strategischer Bedeutung. Als Basistechnologie für Anwendungen in den meisten Bereichen der Wirtschaft und Wissenschaft ist die Informationstechnik (IT) zu einem der wichtigsten Wirtschaftsfaktoren geworden, und darüber hinaus durch seine Auswirkung auf alle Lebensbereiche zu einem bedeutenden gesellschaftlichen und sozialen Faktor. Das Angebot an Software (SW) nimmt zwar immer weiter zu, aber die Stimmen nach Produktivitätssteigerung und Qualitätsverbesserung werden immer lauter. Gegenwärtig wird praktisch in der gesamten Wirtschaft und Gesellschaft eine lebhafte Debatte über einen notwendigen Strukturwandel bzgl. der Produktivitäts- und Qualitäts-Forderungen, die an IT-Produkte gestellt werden, geführt, um eine stärkere Kundenorientierung, erhöhte Flexibilität, Anpassungsfähigkeit zu erzielen und Kosten bei wachsender Qualität zu senken. Bis vor kurzem standen die Anschaffungskosten von Software im Mittelpunkt der Kalkulationen. Heute erkennt man die Nutzungskosten als ein wichtigeres Kriterium für den Einsatz von SW in der betrieblichen Praxis. Geringere Nutzungskosten sind das beste Argument für den Erwerb gebrauchstauglicher Softwareprodukte (SWP) [Dzid/Frei´00].

Softwaregestaltung ist auch Arbeitsgestaltung

Die Struktur eines Softwaresystems, das als Arbeitsmittel genutzt wird, hat einen weitreichenden Einfluss auf die Arbeitsinhalte und Arbeitsabläufe, der keinesfalls auf den Bereich der sogenannten Benutzungsschnittstelle beschränkt ist. *Benutzergerechte Gestaltung von Software* muss in integrativer Weise alle Ebenen eines Mensch-Computer-Systems berücksichtigen; sie muss also von den organisatorischen Auslegungen der Arbeitsaufgaben über die unterstützende Systemfunktionalität bis hin zur konkreten Realisierung in der Benutzerschnittstelle reichen. Eine ganzheitliche Betrachtung der Aufgabenstellung reicht also über das hinaus, was üblicherweise unter Software-Ergonomie gemeint ist [Hack´87].

In der betrieblichen Praxis entsteht Handlungsbedarf bei der Beschaffung von gebrauchstauglicher SW, bei der Qualitätsverbesserung existierender SW, beim Customizing von Standardsoftware sowie bei der Entwicklung von Individualsoftware.

Das Ziel des Usability-Engineering **(UE)** ist es, die Gebrauchstauglichkeit von Softwareprodukten **(SWP)** sicherzustellen, d.h. dass der Nutzer seine Aufgaben mit Hilfe der Software nicht nur erledigen, sondern diese Aufgaben auch effizient (d. h. unter optimalem Aufwand) durchführen kann.

Um die Tauglichkeit der Software für die Nutzung am Arbeitsplatz zu sichern, müssen Software-Engineering **(SE)** und Usability-Engineering einhergehen: Usability-Engineering muss als fester Bestandteil eines modernen SW-Engineering-Prozesses betrachtet werden, damit ein kostengünstigerer SW-Entwicklungs-Prozess durch frühzeitiges Erkennen von Mängeln und eine fachgerechte Standardisierung des Vorgehens gewährleistet werden kann. Zusätzlich braucht der betriebliche Praktiker eine "methodische Ausstattung", um die Tauglichkeit der Software für den Gebrauch am Arbeitsplatz bewerten zu können. Die Methoden ermöglichen auch die Umsetzung von Normen, in Mindestqualität zu sichern. Die Benutzer sollen in

die Lage versetzt werden, ihre Anforderungen genau zu beschreiben, Nutzungsprobleme besser zu begründen und sich an Prozessen zur Verbesserung der Qualität und Produktivität zu beteiligen. Als Grundlage der methodischen Praxis der Softwarebewertung dient in Deutschland das "DATech-Prüfhandbuch Gebrauchstauglichkeit" und der "DATech-Prüfbaustein UE-Prozess Qualität" ([DAT-G'00] [DAT-UE'00]).

Die Innovation

Die Zeiten der Behauptung, dass ein Softwareprodukt an und für sich schon eine Innovation darstelle, sind vorbei. Die Gesetze der Marktwirtschaft [Alba´89], [Sabe´97], [Breu´98], [Breu´00] zwingen uns, die Innovation der Softwareprodukte in ihrer Nutzungs-Qualität zu sehen, um Arbeitsprozesse neu gestalten zu können [Dzid´00]. Künftig wird Gebrauchstauglichkeit von Softwareprodukten ein Wettbewerbsfaktor werden, denn es wird nicht mehr ausreichen, dass Software zuverlässig funktioniert. Der Prozess der SW-Entwicklung sollte darauf vorbereitet sein.

1.1 Einleitung

1.1.1 Ziel der Diplomarbeit

In dieser Diplomarbeit werden die Aspekte der Integration von Usability-Engineering und Software-Engineering untersucht, und es wird ein Versuch zur Bildung einer Brücke zwischen Usability-Engineering und der bisherigen Praxis des Software-Engineering unternommen.

In der Diplomarbeit wird der Fokus auf das V-Modell (Standard für IT-Systeme zur Beschreibung des IT-Entwicklungs-Prozesses) [Drös´00] und das Prozess-Modell für UE, welches im Kapitel 4 modelliert wird, ausgerichtet, um das Einhergehen von Software-Engineering und Usability-Engineering zu untersuchen.

Die zentrale Frage der Diplomarbeit lautet:

> Wie kann das V-Modell um einen Usability-Engineering-Prozess erweitert werden, so dass die Konformität des Vorgehens auch mit der ISO 13407 ("Human centered design processes for interactive systems" [ISO-13407]) gewährleistet ist?

Die Diplomarbeit soll dabei die folgenden Ergebnisse erbringen:

- Darstellung der gegenwärtigen Problemsituation. Es werden die Auswirkungen auf den Software-Entwicklungs-Prozess durch die Nichtbeachtung des UE beschrieben (Kapitel 1.2).

- Ausarbeitung der Aspekte des Usability-Engineering. In diesem Kapitel werden die Eigenschaften, Aufgaben, Methoden (Heuristiken) des UE vorgestellt (Kapitel 2).

- Ausarbeitung der Forderungen in ISO 13407 für den SW-Entwicklungs-Prozess (Kapitel 2).

- Vorstellung des V-Modells und Vergleich mit den Forderungen in ISO 12207 und ISO 13407. Es sollen hier die Defizite des V-Modells bzgl. ISO-Normen aufgezeigt werden (Kapitel 3).

- Erstellung eines Prozess-Modells für ein ISO 13407 konformes UE. Das Modell beschreibt, wie die in ISO 13407 gestellten Forderungen umgesetzt werden können (Kapitel 4).

- Erweiterung des V-Modells um ein ISO 13407 konformes Prozess-Modell für UE. Es werden Vorschläge zur Ergänzung bzw. Modifizierung des V-Modells gemacht, um so eine Grundlage für eine Zertifizierung gemäss ISO 13407 zu schaffen (Kapitel 4).

- Evaluierung der in dieser Diplomarbeit gemachten Vorschläge anhand einer Fallstudie (Kapitel 5).

1.1.2 Ein Prozess-Modell für Usability-Engineering

Die Motivation für den Entwurf eines UE-Prozess-Modells liegt darin, ein Vorgehen zur Sicherung der Gebrauchstauglichkeit bei der Entwicklung von Software zu beschreiben.

Ergebnis dieses Abschnittes ist ein Prozess-Modell für UE, ergänzt um Hinweise für die (exemplarische) Umsetzung in einem Software-Entwicklungs-Projekt. Es soll die bei der SW-Entwicklung einzuhaltenden Arbeitsschritte beschreiben, soweit sie für die Sicherstellung der Gebrauchstauglichkeit (ISO 9241-11) wichtig sind.

Das Vorgehen bei der Erstellung des Prozess-Modells für UE bezieht sich vor allem auf die Klärung der Frage: Welche Arbeitsschritte sind durchzuführen und welche Methodenzuordnung ist dabei anzuwenden?

Zur Erstellung des Prozess-Modells werden die folgenden Schritte unternommen:

- Zunächst wird eine Ausarbeitung der Bestandteile (Aktivitäten und Produkte) des UE-Prozesses vorgenommen. Als wesentliche Grundlage dafür dient [ISO-13407] und [DzidFrei´98].
- Das Ergebnis des UE-Prozesses ist die Anforderungs-Definition, die für den SW-Entwurf bereit zu stellen ist. Der Prozess, der zu Nutzungs-Anforderungen führen soll, muss mindestens folgende Arbeitsschritte beinhalten: Die Erstellung von Kontext-Szenarien, Ableitung von Anforderungen, Erstellung von Use-Szenarien und Use-Cases, das Usability-Prototyping und die Ableitung von Prüfkriterien. Entscheidend für das Prozess-Modell ist, *wie* diese Arbeitsschritte sinnvoll durchzuführen sind, um die Anforderungs-Definitionen zu erhalten. Im Prozess der Anforderungs-Entwicklung, dessen Produkt die Anforderungs-Definition ist, wird ein Prototyping-Prozess durchgeführt, in dem anhand der Use-Szenarien und der Prüfkriterien Merkmale von SWP erprobt werden.

Das Prozess-Modell für UE wird in der für den Regelungsteil des V-Modells typischen Form dokumentiert [Drös'00]: Strukturierte Darstellung des Vorgehens und die Entwicklung von Verfahrensanweisungen.

1.1.3 Erweiterungen des V-Modells um Usability-Engineering

Ziel des Kapitels ist, einen Vorschlag zur Ergänzung bzw. Modifizierung des V-Modells um Elemente des Usability-Engineering vorzulegen. In Verbindung mit den Ergebnissen soll die wesentliche Grundlage für eine Zertifizierung von Entwicklungs-Prozessen nach ISO 13407 anhand des "erweiterten V-Modells" geschaffen werden. Der Vorschlag wird als ein V-Modell konformes Projekthandbuch formuliert.

Die Abgleichung des V-Modells mit dem UE-Modell führt zur Modifizierung/ Erweiterung von folgenden Aktivitäten des V-Modells: System-Anforderungsanalyse (SE1), System-Entwurf (SE2), SW-Anforderungsanalyse (SE3), SW-Grobentwurf (SE4) und des SE.5.1-SW Fehlerbehandlung.

Dabei muss von Anfang an bei der Gestaltung von Software-Entwicklungs-Prozessen das Spannungsfeld zwischen Nutzungskonzept (Entwurf intendierter Nutzung anhand von Szenarien) und Systemkonzept (Entwurf softwaretechnischer Aspekte) in Betracht gezogen werden. Nur so kann für ein Problem eine innovative Lösung gefunden werden, weil man kontinuierlich eine Gegenüberstellung von deskriptiven und konstruktiven Aspekten eines Entwurfs-Prozesses vornimmt.

Schliesslich muss der Entwicklungs-Prozess dahingehend geprüft werden können, ob er den Richtlinien für UE-Prozesse [DAT-UE´00] entspricht. Dies bedeutet zum einen, mit Hilfe von Prozess-Prüfkriterien Abweichungen zu finden, zu bewerten und konstruktive Vorschläge ableiten zu können; zum anderen soll die Zertifizierung des Prozesses im Sinne bestehender Normen ISO 9000-x [Thal´00], [ISO-13407] ermöglicht werden.

1.2 Problemdarstellung

Die ergonomische Gestaltung von Software wird immer noch als ein Randgebiet der SW-Entwicklung verstanden, das hauptsächlich zur Oberflächenkosmetik verkommen ist, und sich wenig mit den Lösungen von aufgabenrelevanten ergonomischen Problemen befasst. Ergonomisch taugliche Lösungen fördern die Effizienz der Erledigung von Arbeitsaufgaben, damit unnötiger Arbeitsaufwand vermieden wird. Der Effizienzmangel seinerseits erschwert die Arbeit und beeinträchtigt die Produktivität. Im Software-Engineering wird die Effektivität der Nutzung einseitig berücksichtigt und den Effizienzanforderungen der Nutzer wird wenig Aufmerksamkeit geschenkt. In der Praxis heisst das, „man ist zufrieden“, wenn die Software korrekt funktioniert; aber ganz ehrlich gesagt, für uns Benutzer: Was nutzt uns eine korrekte Software, wenn wir sie nicht gebrauchen können? Spätestens bei dieser Frage kommt man zu der Erkenntnis, dass *die Validierung* der Software unbedingt vor der Verifikation statt finden muss, und nicht umgekehrt.

1.2.1 Das Synonymen-Problem

Die unterschiedliche Nutzung der Begriffe in der SW-Entwicklung werfen Probleme auf, die das gemeinsame Verstehen (aus der softwaretechnischen und der Benutzer-Sicht) des Nutzungs- und Systemkonzepts erschweren oder sogar unmöglich machen können. Z. B. unter den Begriffen der *Gebrauchstauglichkeit* versteht man im UE (DIN EN ISO 9241-11) die effektive, effiziente und zufriedenstellende Nutzung eines Produkts gemäss den Erfordernissen des Nutzungs-Kontextes. Im Unterschied dazu spricht man im Software-Engineering [ISO-9126] von der *Benutzbarkeit* mit Teilmerkmalen wie Bedienbarkeit, Benutzerfreundlichkeit, Erlernbarkeit, Verständlichkeit. Die Unterschiede in der Terminologienutzung findet man auch in der Verfeinerung der Begriffe: Zum Beispiel ist aus softwaretechnischer Sicht mit der *Effizienz* [ISO-9126] lediglich der Speicherplatzbedarf (für höhere Verarbeitungsgeschwindigkeit) gemeint, nicht jedoch Effizienzanforderungen, die sich aus dem Nutzungskonzept ergeben (ISO 9241-11).

Interessanter gestaltet sich das Problem bei dem Begriff des *Prototypings*. Für einen Systemtechniker gehört dazu in erster Linie ein rein funktionales Prototyping, um den Nachweis der technischen Machbarkeit zu erzielen - unter anderen throw-away (für Demonstrationszwecke, um Erfahrung zu sammeln), rapid, look/feel (Oberflächegestaltung), experimentelles, evolutionäres/inkrementelles, analytisches. Für einen Usability-Engineer dagegen ist das explorative oder konzeptionelle Prototyping entscheidend, welches das Erarbeiten eines gemeinsamen Verständnisses des Produktes und seiner Einsatzbedingungen sowohl aus der softwaretechnischen Sicht als auch aus der Benutzer-Sicht zum Ziel hat (dazu genauer Kapitel 2.2.4.1).

Je nach Quelle finden sich mehr oder weniger einleuchtende Definitionen und Abgrenzungen zu den anderen Begriffen. Wer aber ernsthaft eine benutzerorientierte SW-Entwicklung anstrebt, muss bereits bei der Verwendung der Begriffe darauf achten, sie benutzerorientiert zu interpretieren. Es sei hier auf das Kapitel 2.1 Vereinbarung der Terminologienutzung und vor allem auf das Glossar dieser Diplomarbeit verwiesen.

Die Beteiligung des Benutzers am Entwicklungs-Prozess wurde vor kurzem noch als eine unnötige Belastung angesehen. Die Autoren Gryczan und Züllighoven beschreiben in ihrem Buch [Gryc et al.´93] : “Das traditionelle SW-Engineering hat für die SW-Entwicklung eine Minimierung der Kommunikationsbeziehungen zwischen den beteiligten Gruppen angestrebt...Ähnliche Gedanken stehen hinter der Entwicklung im CASE-Bereich, wo Kommunikation als unnötiger „Overhead“ betrachtet wird [Fort/Norm´92].“ Zu der Entwicklungsstrategie gehört, dass man die Nutzungs-Beschreibung in eine technische Nomenklatur oder in Modelle transformiert, um so Zeit zu sparen. Die Autoren sind der Meinung, dass solch ein Vorgehen verheerende Folgen für die Validität des Nutzungs-Kontextes hat, weil es ihn meist verfälscht. Ob man z. B. ER-Modelle (Entity-Relationship-Modell) oder Jacobson-System-Design benutzt, benötigt man immer eine Darstellungstechnik, die Sachverhalte des Nutzungskontextes in eine formale Sprache transformiert.

In der Veröffentlichung [Floy et.al´92] im Abschnitt „On Controllability“ verweist auch Dzida darauf und äussert: „Furthermore, methods such as SADT, PSL/PSA result in a representation of the application field reflecting the characteristics of the method applied. Thus, the model of reality is biased by leaning the representation towards the methodical requirements of system design or system specification.“ (Seite 130). Um diesem Dilemma beizu-

kommen sind Szenarien und Prototypen wichtige Mittel, um die Nutzung des künftigen SWP möglichst unverfälscht zu analysieren und unvoreingenommen zu betrachten.

1.2.2 Probleme bei Vernachlässigung der Nutzungs-Qualität von Software

Software-Nutzung ist ein schwer zu verstehender Sachverhalt, weil meist die NutzungsQualität aus unterschiedlichen Sichten betrachtet wird. Zum Beispiel früher dominierte die Effektivitäts-Sicht d.h. die Genauigkeit und Vollständigkeit, mit der die Software das gewünschte Ergebnis lieferte; dabei war die Effizienz der Nutzung lange Zeit nebensächlich - zu lange, wie Produktivitätsuntersuchungen heute deutlich zeigen [Land´95].

Die **Effizienz der Nutzung** ist ein unterschiedlich motiviertes Konzept, je nachdem, was man in Rechnung stellt: Bei gegebener Effektivität nimmt man Kosten (Anschaffung, Training, Wartung) für die Berechnung der *Nützlichkeit*, Zeit für Berechnung der *Produktivität*, und Aufwand für Berechnung der *Gebrauchstauglichkeit* [Dzid/Frei´00]. Der Schlüssel zur Steigerung der Nutzungs-Qualität liegt in der Verbesserung der Gebrauchstauglichkeit der Software, was heisst, dass die ergonomische Qualität eine kalkulierbare betriebswirtschaftliche Grösse ist [Bias/Mayh´94. Um einen einfachen Nachweis für diese Behauptung zu führen, braucht man nur den vom Benutzer zu leistenden Aufwand an Arbeitsschritten in Ansatz zu bringen (bei gegebener Effektivität). Dazu ein Rechenbeispiel [Dzid´99]: „Angenommen, 250 Arbeitskräfte in einer Versicherungsfirma bearbeiten jeweils 27 Masken an 220 Arbeitstagen im Jahr. Eine ergonomische Verbesserung trage dazu bei, dass nur 1 Sekunde pro Maske eingespart wird. Setzen wir nur DM 25 Gehalt pro Stunde an, so ergibt sich unter diesen Annahmen eine Einsparung von DM 10.312 pro Jahr (250x 27x 220x 1/3600x 25). Nun ist aus der ergonomischen Gestaltungspraxis bekannt, dass es den Arbeitsaufwand nicht nur in Sekunden verringen hilft. Bei einer Zeiteinsparung von nur 10 Sekunden spart der Anwender bereits DM 100.000; bei einer Minute kommt mehr als eine Million heraus (vielleicht mehr als der Anschaffungspreis der Software). In dieser Kalkulation sind nicht einmal die durch ergonomische Gestaltung verringerten Nutzungsfehler und Kosteneinsparungen für Schulung und Benutzerbetreuung berücksichtigt“.

Ein SW-Entwickler wäre überfordert, wenn er sämtliche Nutzungsprobleme voraussehen wollte, aber er sollte im Stande sein, notwendige Massnahmen ergreifen zu können. Für Software gilt, dass sie nicht frei von Fehlern ist, aber auch nicht von ergonomischen Mängeln [Müll/West´94]. Viele *Nutzungsprobleme* werden erst nach der Einarbeitungsphase an den Arbeitsplätzen der Benutzer festgestellt. Es bedarf deshalb eines Pflegeprozesses, um die ergonomischen Mängel spätestens beim nächsten Release beseitigen zu können. Die Nutzungs-Qualität ist folglich nicht nur am gelieferten Produkt festzumachen, sondern auch am zugrunde liegenden Entwicklungs- und Pflege-Prozes*s*. Damit man künftig gebrauchstaugliche SWP produzieren kann, muss man den Schwerpunkt der Qualitäts-Sicherung (QS) ins Vorfeld des Entwicklungs-Prozesses verlegen [Dzid´00], nämlich in die Planung der SW-Entwicklung (genauer siehe Kapitel 4). Es ist wichtig, den Nutzungs-Kontext eines SWP bereits vor der Entwicklung sorgfältig analysiert zu haben, was eine Aufgabe des Usability-Engineering ist.

Zusammenfassend kann man sagen, dass die Vernachlässigung des Nutzungskonzeptes der Software zu unfreiwilligem, zusätzlichem und kostspieligem Arbeitsaufwand führt, indem die neuen Aufgaben bewältigt werden müssen, während man die Zeit mit unnötigen Nutzungsproblemen verschwendet. Der erwartete Rationalisierungs-Effekt bleibt aus!

1.2.3 Die Kosten und das Problem mit der Produktivität

Software-ergonomische Probleme sind nicht zu unterschätzen, denn sie erschweren die Nutzung von Software und behindern den Arbeitsprozess enorm. Ein schwerwiegender Mangel verhindert sogar die Erreichung des Arbeitsergebnisses, was hohe Kosten verursacht [Mina´00] (siehe Effektivitätsmangel [ISO-9241-11]).

Wenn man hier über die Kosten spricht, muss man nicht nur die Anschaffungskosten, sondern vor allem die Nutzungskosten berücksichtigen. Nutzungskosten sind Kosten z. B. für Wartung, Customizing, Systemverwaltung, Schulung und Benutzerbetreuung. Diese Kosten (Produktivitätsverluste nicht eingerechnet) sind im allgemeinen dreimal so hoch wie die Anschaffungskosten. Solche Folgekosten können durch gebrauchstaugliche Software erheblich gesenkt werden. Die Beachtung ergonomischer Kriterien bei der SW-Entwicklung ist damit auch ein Beitrag zur Rationalisierung! [Dzid/Frei´00]

Neben der Kostenreduktion bei den Anwendern und Benutzern ist auch die Kostenersparnis auf Seiten der SW-Entwicklung zu sehen. Auch wenn die ergonomisch orientierte SW-Entwicklung zunächst einen zusätzlichen Aufwand bedeutet (z. B. Erstellung von Szenarien, aufgabenorientierten Anforderungen, Prototyping), so muss man auch sehen, dass an anderer Stelle erhebliche Kosten eingespart werden: Wenn Mängel durch adäquates Vorgehen beim SW-Entwurf frühzeitig entdeckt werden, kostet deren Behebung erheblich weniger Geld als wenn sie bei der Implementierung beseitigt werden müssten. Es ist nachgewiesen [Land´95], dass Mängelbeseitigung in früheren Entwicklungsphasen erheblich billiger ist. Die Kosten für jeden aufgedeckten Fehler betragen beim Entwurf das einfache, bei der Implementierung das sechsfache und bei der Wartung und Pflege das zehnfache.

Einige Quellen zum Thema Produktivität bestätigen, dass man durch den Einfluss des Usability-Engineering bei der Gestaltung von Software Zeiteinsparungen und Kostenreduktion vorweisen kann [Bias/Mayh´94]. Dies zeigt sich in besserer Integration von Anwendungsprogrammen, verbesserter Einstellung der Benutzer zum SWP und geringeren Kosten für Benutzerschulung, geringerer Wartung und Pflegekosten, Verbesserung der Bedingungen im Nutzungs-Kontext, Verringerung der Nutzungsfehler um das Fünffache, Produktivitätssteigerung [Land´95]. Andererseits sieht Landauer "im ausgebliebenen Beitrag zur Produktivität der IT-Branche ein Zeichen ihrer Unreife und hofft auf besser benutzbare SWP der Zukunft". Weiter wird Landauer in iX'3/2000 auf Seite 34 zitiert: "Trotz überall verfügbarer PCs ist eine aussergewöhnliche Zunahme der Arbeitsproduktivität nicht erkennbar. Zu den Ursachen des nachlassenden Produktivitätsfortschrittes gehört möglicherweise die IT selbst".

Die Abbildung 1-1 zeigt den Verlauf des Produktivitätsfortschrittes über die Jahre in verschiedenen Branchen. Es ist nachgewiesen, dass die erwartete Produktivitätssteigerung in Dienstleistungsbereich nicht erfolgt ist. Ab 1973, dem statistisch gesetzten Beginn der Computerisierung der Dienstleistungen, sind die Produktivitätszuwächse rückläufig, während die Computerisierung in der Fertigungsindustrie (manufacturing) steigende Zuwächse aufweist.

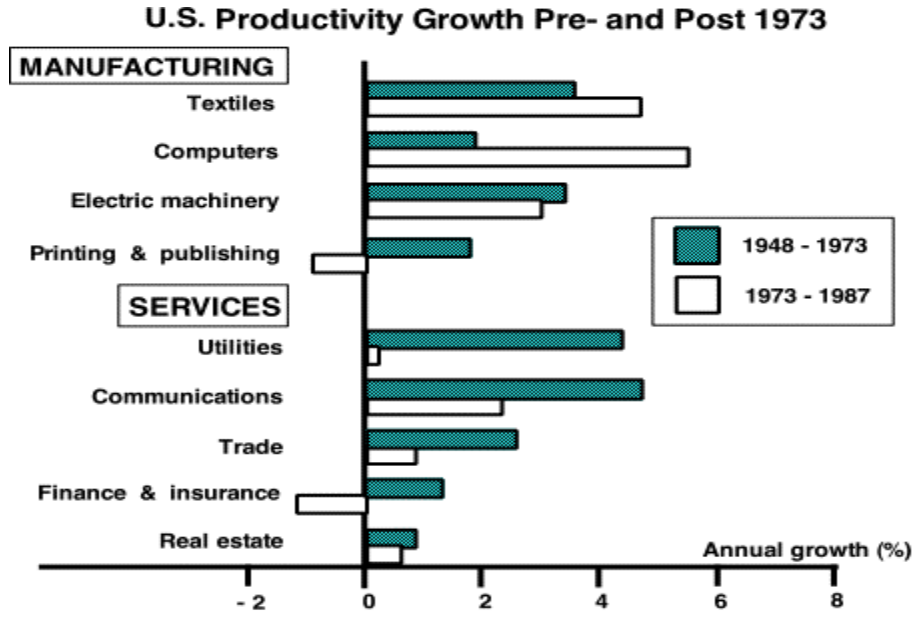

Abbildung 1-1 Der Produktivitätsfortschritt? [Dzid/Frei´00]

Die IT-Industrie hat es bis heute vernachlässigt, sich mit den Gebrauchstauglichkeitsfragen bei der Gestaltung von Software auseinander zu setzen [Beim et.al.´93]. In der Regel überlässt man das Problem dem Anwender, der seinerseits aber, keine oder wenig Kenntnisse aus der softwaretechnischen Sicht besitzt und eine Chance auf Verbesserung seiner Arbeitsprozesse gar nicht erst in Betracht zieht. Das hat zur Folge, dass die Gebrauchstauglichkeit kaum berücksichtigt wird.

1.2.4 Die Bildschirmarbeitsverordnung und die ISO-Normen

Bildschirmarbeitsverordnung

Um Softwareprodukte für die Nutzung an Arbeitsplätzen tauglicher zu machen, wurde eine Reihe internationaler Normen (ISO 9000, ISO 9126, ISO 9241, ISO 13407) verabschiedet. Zusätzlich wurden gesetzliche Vorschriften (Bildschirmarbeitsverordnung) in Kraft gesetzt.

Mit der EG-Bildschirmrichtlinie wurde europaweit angestrebt, einheitliche Mindest-Bedingungen für die Arbeit an Bildschirmarbeitsplätzen zu erreichen. Diese Richtlinie wurde inzwischen von allen Mitgliedsländern in nationales Recht umgesetzt.

In Deutschland gilt die Bildschirmarbeitsverordnung [URL:bma] in Verbindung mit dem Arbeitsschutzgesetz. Der Gesetzgeber hat mit der Bildschirmarbeitsverordnung eine klare Aussage zur Einhaltung von Mindestanforderungen getroffen. Diese Verordnung verpflichtet die Anwender, in ihrer Rolle als Arbeitgeber, dafür zu sorgen, dass am Arbeitsplatz verwendete Software den Grundsätzen der Software-Ergonomie entspricht, d.h. die Software konform zu den ISO Normen (im wesentlichen [ISO-9241] und [ISO-13407]) entwickelt wurde. Die Hauptforderung darin lautet: "Die Grundsätze der Software-Ergonomie sind anzuwen-

den". Es liegt nahe, die mit diesen Grundsätzen verbundenen Forderungen zu erfüllen, insbesondere die in der Norm [ISO-9241] Teil 10.

Viele dieser Forderungen wurden allgemein formuliert, aus diesem Grunde sind sie immer im Zusammenhang mit den Erfordernissen des Nutzungs-Kontextes eines Produktes zu interpretieren und zu präzisieren.

Die Konformität mit software-ergonomischen Normen wird als ein fachlich qualifizierter Indikator dafür angesehen, dass von einem Softwareprodukt keine unnötige Belastungen mindestens im Sinne der Bildschirmarbeitsverordnung ausgehen. Eine solche Konformität können sich Anwender vom Hersteller zusichern lassen [DIN EN 45014].

Die Einhaltung der ISO Normen

Während Hardware schon seit langem geprüft wird und sich die Hersteller von Büroeinrichtungen auf DIN berufen, fällt es den Softwareentwicklern immer noch schwer, sich an ergonomischen Normen zu orientieren.

Die heutige Informatik bietet noch keine Lösung, wie die bestehenden ergonomischen Normen (vor allem [ISO-9241] und [ISO-13407]) in einem SW-Engineering-Prozess praktisch umzusetzen sind, damit gebrauchstaugliche Software produziert werden kann. Um diesem Defizit abzuhelfen und die Tauglichkeit der Software für den Arbeitsplatz des Benutzers zu verbessern, müssen Software-Engineering und Usability-Engineering einhergehen: Usability-Engineering (genauer im Kapitel 2 und 4) muss als fester Bestandteil in einen Software-Engineering-Prozess integriert werden.

Die Normreihe [ISO-9241] „Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten“: Die "International Organization for Standardization" hat in [ISO-9241] verschiedene Teile dieser Norm entwickelt:

- Teil 1: General Introduction
- Teil 2: Guidance on task requirements
- Teil 3: Visual display requirements
- Teil 4: Keyboard requirements
- Teil 5: Workstation layout and postural requirements
- Teil 6: Environmental requirements
- Teil 7: Display requirements with reflections
- Teil 8: Requirements for displayed colors
- Teil 9: Requirements for non-keyboard input devices
- Teil 10: Dialogue principles
- Teil 11: Usability statements
- Teil 12: Presentation of information
- Teil 13: User guidance
- Teil 14: Menu dialogues
- Teil 15 Command dialogues
- Teil 16: Direct manipulation dialogues
- Teil 17: Form filling dialogues

Die einzelnen Teile der Norm ergänzen einander. So muss zum Beispiel ein Programm, das beansprucht gebrauchstauglich zu sein, im Einzelfall auch taugliche Farbkombinationen (siehe Teil 8) berücksichtigen. In der Abbildung1-2 wird der Zusammenhang der für die Softwaregestaltung relevanten Teil-Normen dargestellt.

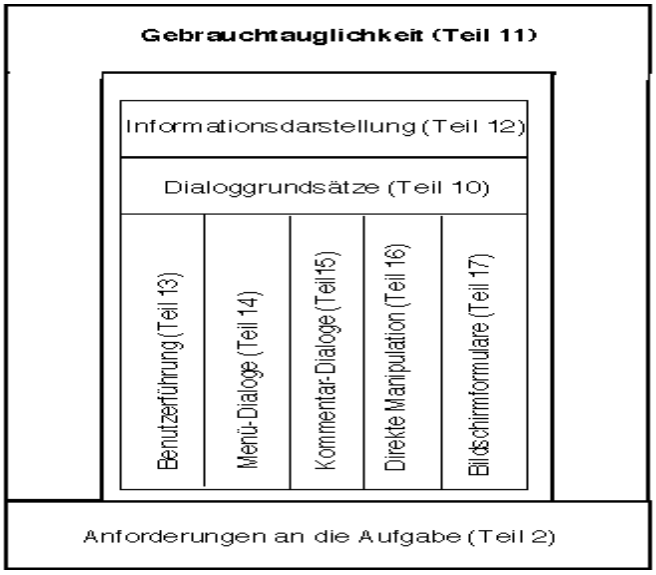

Abbildung 1-2 Zusammenhang der relevanten Teil-Normen für Softwaregestaltung

Die Gebrauchstauglichkeit ([ISO-9241-11]) setzt Benutzbarkeit voraus ([ISO-12119], [ISO-9126]). Während Benutzbarkeit den Aspekt der Effizienz beschreibt, der durch kontextneutrale Merkmale eines SWP realisiert wird, zielt die Gebrauchstauglichkeit auf den Aspekt der Effizienz, der durch ausführbare Tätigkeiten im Kontext nachzuweisen ist. Im Kapitel 2 wird genauer auf die Definitionen zur Gebrauchstauglichkeit, Effektivität und Effizienz eingegangen. In der Abbildung1-3 wird die Struktur von ISO 9241/10-17 dargestellt.

Abbildung 1-3 Struktur von ISO 9241, Teile10-17 [Dzid/Frei´00]

Die Norm [ISO-9126] für Produktqualität
Sie beschreibt die Qualitätsmerkmale und Teilmerkmale eines Softwareprodukts. Die Qualitätseigenschaften setzen sich zusammen aus:

- *Funktionalität*: Angemessenheit, Eignung, Erfüllung, Genauigkeit (Sorgfalt, Präzision), Interoperabilität, Ordnungsmässigkeit, Richtigkeit, Sicherheit (Daten).
- *Zuverlässigkeit*: Fehlertoleranz, Reife, Wiederherstellbarkeit.
- *Benutzbarkeit*: Bedienbarkeit, Erlernbarkeit, Verständlichkeit.
- *Effizienz*: Zeitverhalten, Verbrauchsverhalten, Ressourcenbeanspruchung.
- *Änderbarkeit*: Analysierbarkeit, Modifizierbarkeit, Prüfbarkeit, Stabilität, Wartbarkeit.
- *Übertragbarkeit*: Adaptierbarkeit, Anpassbarkeit, Austauschbarkeit, Ersetzbarkeit, Installierbarkeit, Konformität, Portabilität.

Neben den funktionalen Anforderungen an das SWP sind die nicht-funktionalen Qualitätsforderungen zu erheben. Nur wenn von den hier geforderten Produkteigenschaften ausgegangen wird, können QS-Massnahmen – allerdings rein software-technischer Natur - adäquat gestaltet werden; der Aufwand und die Kosten müssen gerechtfertigt sein.

Durch diese Norm werden die Anforderungen des Qualitäts-Managements (QM) realisiert (siehe ISO 9000-3).

An dieser Stelle sei ein Hinweis, dass die weiteren Normen [ISO-13407], [ISO-12207] in Kapiteln 2 und 3 ausführlich behandelt werden.

Zusammenfassend ist es aus folgenden Gründen notwendig, ergonomische Vorgehensweisen bei der Entwicklung von SWP zu berücksichtigen:

⇒ Gesetzliche Vorschriften (Bildschirmarbeitsverordnung)
⇒ Nutzungsvorteile (damit verbunden Kostenvorteile) bei Anwendern
⇒ Kostenersparnis bei Entwicklern/ Anwendern

Zudem erweist sich die Gebrauchstauglichkeit von Softwareprodukten zunehmend als ein Wettbewerbsfaktor für Software-Entwickler, weil immer mehr Anwender Wert darauf legen, norm-konforme Softwareprodukte zu erwerben.

Qualitätssicherung mit Hilfe des Usability-Engineering

Die oben vorgestellten ISO Normen zur Sicherung der Qualität des SWP zeigen ein Defizit bezüglich der Qualität der Nutzung des SWP auf. In folgender Abbildung 1-4 wird die fehlende Berücksichtigung des UE-Prozesses im QM aufgezeigt - der gestrichelte Pfeil zeigt den fehlenden Einfluss des heutigen QM auf die Nutzungs-Qualität.

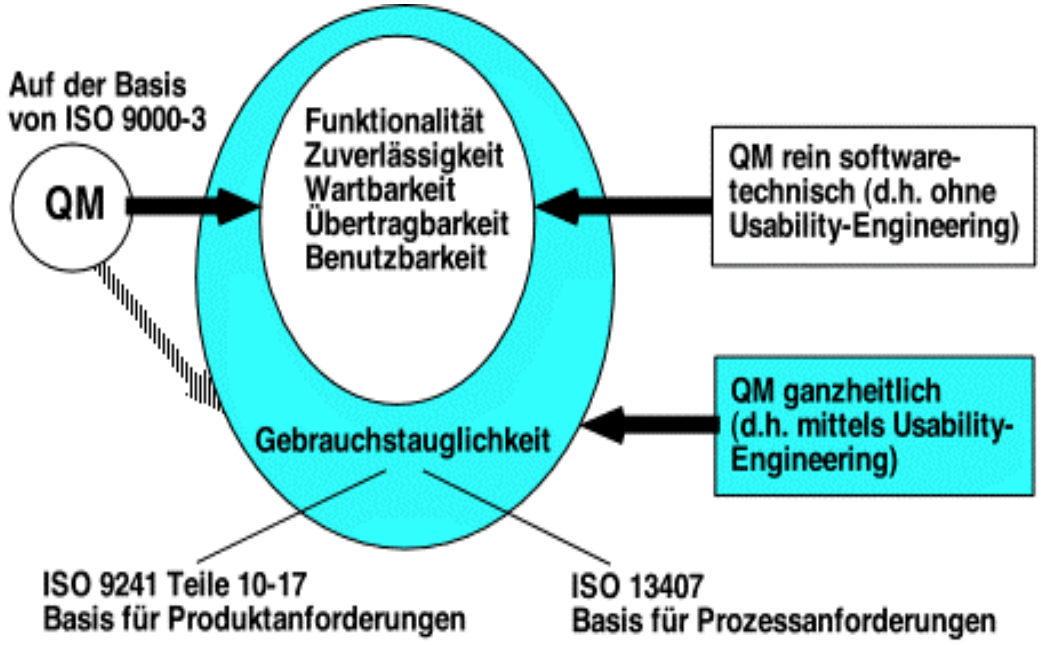

Abbildung 1-4 Ganzheitliches Qualitätsmanagement mittels UE [DATech-UE´00]

In dieser Abbildung ist das Usability-Engineering in den Gesamtprozess der Qualitätssicherung (QS) eingebettet und der Zusammenhang der relevanten ISO-Normen, die in dieser Diplom-Arbeit eine entscheidende Rolle spielen, deutlich gemacht.

In einem UE-Prozess werden die Voraussetzungen dafür geschaffen, das ein gebrauchstaugliches Produkt im Sinne [ISO-9241-11] und [ISO-9126] entsteht. Denn heute genügt nicht mehr ein allein auf software-technische QS gerichteter Prozess den Anforderungen der Anwender.

2 Das Usability-Engineering

In diesem Kapitel wird der Begriff Usability-Engineering und seine Aufgaben bei der Software-Entwicklung vorgestellt. Usability-Engineering umfasst sowohl Entwicklungsaktivitäten (z.B. Anforderungs-Entwicklung und Prototyping) als auch Massnahmen der softwareergonomischen Qualitätssicherung. Methoden des UE und Massnahmen der QS lassen sich kaum im Einzelnen in einer Norm festlegen, weil sich Software-Entwicklungsprojekte stark unterscheiden und es dem Hersteller überlassen bleiben sollte, welche Methoden unter den gegebenen Umständen eingesetzt werden.

2.1 Vereinbarung der Terminologienutzung

In der heutigen Informatik werden keine Vorschläge gemacht, wie die software-technischen Aspekte mit den Usability-Aspekten in der Software-Entwicklung zusammen zu führen sind. Im Grunde genommen kündigt sich schon die bis heute oft unerkannt gebliebene Spaltung der beiden Bereiche bei der Begriffs-Semantik an, die kurz schon im Kapitel 1.3 angesprochen wurde. Ausserdem ist in vielen Fällen der Begriff des *Nutzungs-Kontextes* (Nutzungs-Qualität ist relativ zum Kontext), für den ein Produkt entwickelt wird (oder wurde), nicht genau beschrieben. Bei der Prüfung der *Benutzbarkeit* nach [ISO-12119] wird deshalb kein spezieller (konkreter), sondern ein *intendierter Nutzungs-Kontext* zu Grunde gelegt, den der Hersteller beschreiben muss; z.B. hinsichtlich der beabsichtigten Aufgaben, die man mit dem Softwareprodukt bearbeiten können soll. In realen Nutzungs-Kontexten mögen diese vom Hersteller beschriebenen Aufgaben jeweils etwas anders aussehen, so dass ein Anpassungsproblem entsteht, das mittels Customizing zu lösen versucht wird. Für ein Standardprodukt wird es viele solcher Anpassungsprojekte geben. Es wäre jedoch für den Hersteller schwierig, Prüfungen auf Gebrauchstauglichkeit in den vielen verschiedenen Nutzungs-Kontexten durchführen zu lassen. Das DATech Prüfhandbuch schlägt vor: „Allerdings wäre es möglich, aus den vielen Anpassungsprojekten allmählich ein Produkt entstehen zu lassen, das immer besser auf die spezifischen Anforderungen der Nutzungs-Kontexte abgestimmt ist. Wenn nachvollziehbar ist, dass viele kontextspezifischen Anpassungen bei einem Release weitgehend berücksichtigt wurden, so wäre es möglich, ein Produkt hinsichtlich verschiedener Nutzungs-Kontexte auf Gebrauchstauglichkeit zu prüfen. Die Gebrauchstauglichkeitsprüfung wäre dann grundsätzlich auch beim Hersteller möglich. Der Hersteller müsste nachweisen, dass Anpassungswünsche aus vielen verschiedenen Nutzungs-Kontexten zu einem neuen Release geführt haben. Allerdings müssten typische Benutzer aus verschiedenen Nutzungs-Kontexten in die Zufriedenstellungsbeurteilung einbezogen werden.“ [DAT-G´00], Seite 25-26

Um die richtige Verwendung der Begriffe zu gewährleisten und somit Verständigungsprobleme zu vermeiden, werden im Folgenden die wichtigsten Begriffe vorab vorgestellt, die im UE-Bereich gelten, (vor allem [DAT-UE'00] [DAT-G'00] [ISO-13407-2] [Niel'93] [Paul et al.'95] [Thal'00]). Die restlichen Begriffe, die in dieser Arbeit benutzt wurden, sind im Glossar aufgelistet (durch > gekennzeichnet):

Anforderung (requirement) auch Nutzungsanforderung: Ein qualitativer oder quantitativer Wert einer Qualitätsforderung, die aus >Erfordernissen der Arbeit oder Organisation (>Nutzungs-Kontext) des Benutzers sowie seiner Belange abgeleitet ist. Anforderungen sind i.d.R. als geforderte >Arbeitstätigkeit zu formulieren. Nur wenn gesicherte ergonomische Erkennt-

nisse für die Merkmale eines Produkts (z.B. Farbkodierung, Hintergrundfarbe des Bildschirms) gegeben sind, können diese Merkmale als Anforderung dienen. Eine Anforderung kann in ein >Prüfkriterium transformiert werden, wenn die Tätigkeitsforderung für die Interpretation (Präzisierung) einer genormten Anforderung [ISO-9241-10] benutzt wird.

Benutzbarkeit (usability): Eine Menge von Merkmalen zur Erleichterung der Benutzung mit Rücksicht auf die individuelle Bewertung durch eine festgelegte oder vorausgesetzte Gruppe von >Benutzern. Benutzer (user /end user) Eine Person oder eine Personengruppe, die Rechner zur Unterstützung ihrer Aufgabenbearbeitung einsetzt.

Design-Use-Cycle: Die Phasen, die ein Softwareprodukt durchläuft, beginnend mit der Produktidee über die Entwicklung des Nutzungs-Konzepts, über den Entwurf, die Implementierung und die Installierung des Produkts beim Anwender, bis hin zur >Pflege des Produkts im Nutzungs-Kontext, in dem aufgrund der Nutzungserfahrungen das nächste Release vorbereitet wird. Eine iterative Folge der Phasen ist üblich. Entwicklungsaktivitäten und Massnahmen der >Qualitätssicherung wechseln sich ab.

Effizienz (efficiency): Der im Verhältnis zur Genauigkeit und Vollständigkeit (Effektivität) eingesetzte Aufwand, mit dem >Benutzer ein bestimmtes Ziel erreichen [ISO 9241-11].

Erfordernis (implied need): Eigenschaft der Arbeit oder Organisation (>Nutzungs-Kontext), die selbstverständlich als Forderung vorausgesetzt wird und aus der sich spezifische >Anforderungen ableiten lassen.

Gebrauchstauglichkeit (usability within context of use): Das Ausmass, in dem ein Produkt durch bestimmte >Benutzer in einem bestimmten >Nutzungs-Kontext genutzt werden kann, um bestimmte Ziele >effektiv, >effizient und >zufriedenstellend zu erreichen (ISO 9241-11).

Kontext-Szenario (context scenario): (methodisches Vorgehen um den Arbeitsprozess zu beschreiben). Eine episodische Beschreibung der >Arbeitstätigkeiten einer Arbeitsperson (z.B. der künftige >Benutzer), um den Einfluss der Bedingungen des >Nutzungs-Kontexts auf die Ausführung der Tätigkeiten zu analysieren. Diese Erhebung dient der Vorbereitung der Anforderungs-Entwicklung zu Beginn eines Projekts. Es werden in der Sprache des Benutzers die Nutzungsbedingungen eines noch zu entwickelnden Systems beschrieben, um ein klares Verständnis der >Erfordernisse der Arbeit und Organisation zu erarbeiten. Die sachliche Richtigkeit und Vollständigkeit der Beschreibung muss vom Benutzer bestätigt werden (Validierung).

Nutzungs-Kontext (context of use): Faktoren, die die >Gebrauchstauglichkeit eines Produktes beeinflussen können, z.B. die Qualifikation des >Benutzers, die Merkmale des Arbeitsinhalts, der Arbeitsplanung, der Arbeitsteilung sowie die physische und soziale Umgebung (vgl. [ISO 9241-11]).

Prototyping (prototyping): Ein iterativer Prozess zur Herstellung von Teilen einer >Benutzungsschnittstelle anhand vom >Use-Szenarien sowie der Herstellung oder Simulation einer dahinter liegenden Funktionalität zum Zwecke der Veranschaulichung der vom >Designer verstandenen >Nutzungsanforderungen. Dient ein >Prototyp der Erkundung noch nicht verstandener Nutzungsanforderungen, so nennt man diesen Prozess exploratives Prototyping.

Usability-Engineering (usability engineering): Entwicklung von >Nutzungsanforderungen, >Prototypen und Softwareprodukten, deren Validierung und Verbesserung, unter systematischer Anwendung von >Usability-Engineering-Methoden im Rahmen eines >Design-Use-Cycle erfolgt.

Usability-Engineering-Methoden (usability-engineering methods): Methoden zur Entwicklung von innovativen Nutzungs-Konzepten, >Nutzungsanforderungen und >Prototypen, deren Validierung, Bewertung und schrittweise Verbesserung während des Entwurfs- und >Pflegeprozesses.

Use-case: Die Kurzform eines >Use-Szenarios. Jede Interaktion des >Benutzers mit dem Dialogsystem (>Dialogschritt) wird in zwei Spalten festgehalten, links die >Arbeitstätigkeit des Benutzers, rechts die darauf folgende Reaktion des Dialogsystems.

Use-Szenario (use scenario): Eine episodische Beschreibung von >Arbeitstätigkeiten, die ein >Benutzer zur Erledigung einer Kern-Aufgabe am Dialogsystem erledigt. Diese Erhebung dient der Anforderungs-Entwicklung während des Prototyping oder der Ermittlung von Anforderungen am fertigen Produkt zum Zwecke der Softwareprüfung. Während der Softwareprüfung an einem existierenden System braucht die Interaktion mit dem Dialogsystem nicht ausführlich erfasst zu werden; es reicht eine Erfassung in Form eines >Use-case aus. >Dialogschritte, die jedoch auf Nutzungsprobleme oder auf vermutete >Normabweichungen hinweisen (critical incidents), sind ausführlich zu beschreiben und anschliessend auszuwerten (Prüfung der >Konformität).

Zufriedenstellung (user satisfaction): Beeinträchtigungsfreiheit der Nutzung und positive Einstellungen der >Benutzer zur Produktnutzung [ISO-9241-11].

2.2 Aspekte des Usability-Engineering

In diesem Kapitel wird der Bereich des Usability-Engineerings vorgestellt, angefangen mit der Klärung des Usability-Begriffs und des Nutzungs-Konzepts, über Aufgaben des UE, Anforderungs-Entwicklung bis hin zu UE-Heuristiken.

Der Begriff Usability

Nielsen [Niel´93] definiert Usability zum Beispiel durch folgende Begriffe:

- *Lernförderlichkeit* [User: Novices]: schneller Einstieg in ein neues System (gut messbar mit unerfahrenen Benutzer), Ausnahmen: Experten-Systeme;
- *Effizienz* [User: PowerUsers]: hohe Produktivität soll möglich sein. Nach der Lernphase Zeit für Aufgabenbewältigung (messbar mit erfahrenen Benutzer);
- *Wiedererkennbarkeit* (Memorability) [User: CasualUsers]: Benutzer sollen nicht jedesmal neu lernen müssen; wichtig für Tools, Steuerungsprogramme;
- *Fehler*: möglichst wenige, wiedergutzumachende Fehlerrate des Benutzer (messbar);
- *Zufriedenheit*: Akzeptanz.

Im DATech-Prüfhandbuch dagegen wird die Usability (Effizienz) durch folgende Forderungen ausgezeichnet [DAT-G'00]:

- **Aufgabenangemessenheit:** Ein SW-Programm ist „aufgabenangemessen, wenn es zur Erledigung Ihrer konkreten Tätigkeit brauchbar ist. Brauchbar bedeutet, dass alle Tätigkeiten, die Benutzer erledigen müssen, vom Programm unterstützt werden und das Programm dabei wirklich eine Hilfe und kein nötiges Übel ist, das die Arbeit in manchen Situationen eher erschwert oder umständlicher macht“.
- **Selbstbeschreibungsfähigkeit**: ein SW-Programm ist „selbstbeschreibungsfähig, wenn Benutzer jederzeit informiert sind, was der Computer gerade macht und was er als nächs-

tes als Eingabe oder Reaktion erwartet. Dies bedeutet unter anderem, dass Benutzer alle Rückmeldungen verstehen können, immer wissen, wo als nächstes etwas eingeben müssen und sich jederzeit klar über die Folgen sind, die eine Eingabe haben wird“.

- **Erwartungskonformität**: ein SW-Programm ist „erwartungskonform, wenn Benutzer bei der Arbeit mit dem Computer keine "Überraschungsmomente" erleben. Solche Momente können zum Beispiel sein, wenn sich eine Funktion an einer ganz anderen Stelle im Menü befindet, als man gedacht hätte, oder wenn Aufgaben nicht wie gewohnt ausgeführt werden können“.
- **Lernförderlichkeit**: ein SW-Programm ist „lernförderlich, wenn es Benutzer unter anderem ermöglicht, selbständig einfach mal "rumzuprobieren", ohne dass man Angst haben muss, etwas "kaputt" zu machen. Zusätzlich sollten Benutzer durch das Programm die für sie relevanten Informationen erhalten, die ihrer Meinung nach benötigt werden, um das Programm besser zu verstehen“.
- **Steuerbarkeit**: ein SW-Programm ist „steuerbar, wenn Benutzer die Abfolge der Arbeitsschritte weitgehend selbst bestimmen können. Wenn es die Arbeitssituation erfordert, kann man die Arbeit am Computer unterbrechen und diese dann ohne Verlust der bis dahin erreichten Arbeitsergebnisse wieder aufnehmen".
- **Fehlertoleranz**: ein SW-Programm ist „fehlertolerant, wenn Benutzer ihre Arbeitsergebnis trotz fehlerhafter Eingaben entweder mit keinem oder mit minimalem Korrekturaufwand erreichen können. Dies bedeutet, dass es durchaus erlaubt sein muss, sich zu vertippen oder einen falschen Arbeitsschritt zu machen, ohne dass das Programm gleich abstürzt, oder man den Fehler nur mit Mühe wieder gut machen kann. Ausserdem sollte das Programm darauf aufmerksam machen, wenn es einen Fehler bemerkt, und Ihnen mögliche Korrekturhinweise liefern“.
- **Individualisierbarkeit**: ein SW-Programm ist „individualisierbar, wenn Benutzer Einstellungen des Programms an ihre individuellen Bedürfnisse anpassen können“.

Oft wird unter Usability nur ergonomische Bewertung verstanden – das ist leider zu wenig, um professionell Usability-Engineering zu betreiben, denn es fehlt an konstruktiven Vorschlägen und Methoden zur ihrer Durchführung. Die konstruktiven Vorschläge dafür werden ab dem Absatz „Das Usability-Engineering“, aber vor allem im Kapitel 4 „Ein Prozess-Modell für UE“ vorgestellt.

Das Usability-Engineering

Per Definition umfasst das Usability-Engineering die Entwicklung von Nutzungs-Anforderungen, Prototypen und Softwareprodukten, deren Validierung und Verbesserung unter systematischer Anwendung von UE-Methoden im Rahmen eines Design-Use-Cycle erfolgt.

Usability-Engineering ist ein ausgewogenes Zusammenspiel zwischen Benutzern, Fachabteilungen, Usability-Team (Usability-Analytiker, Usability-Engineer, Usability-Prüfer), Ergonomie-Experten und Systementwicklern – dies kann, in kleineren Projekten, eine Gruppe von zwei, drei Personen erfüllen. Sie müssen unter Einbeziehung bestehender Normen und Richtlinien, Industriestandards, Styleguides, Analyse-Methoden, Prototyping (-Methoden, -Werkzeuge) und ausgewählten Evaluationsmechanismen die System- und Nutzungs-Anforderungen gemeinsam entwickeln und definieren, Fachspezifikationen sind zu erarbeiten und daraus eine benutzerorientierte Systemgestaltung vorzubereiten. Die benutzerorientierte Systemgestaltung muss sich an den Anforderungen und Fähigkeiten des Benutzer ausrichten

und setzt deshalb die Einbeziehung des Benutzers in den Gestaltungsprozess voraus. Denn nur der Benutzer selbst verfügt über hinreichende Kenntnis der konkreten Arbeitsabläufe und Arbeitssituationen, in die das zu entwickelnde System bestmöglich eingepasst sein sollte, um die Arbeitsprozesse effizienter durchführen zu können und die Nutzungskosten zu senken. Die Nutzungskosten können dann gesenkt werden, wenn die Ursachen für die Nutzungsprobleme besser erkannt und beseitigt werden. Sie liegen oft darin, das organisatorische, technische oder aufgabenspezifische Erfordernisse bei der Entwicklung von Software nicht ausreichend berücksichtigt werden. Um das zu vermeiden, müssen Benutzer in die Bewertung von Softwarequalität einbezogen werden. Durch den Ansatz einer benutzerorientierten Systemgestaltung eröffnet sich die Möglichkeit, z. B. eingefahrene und überholte Arbeitsabläufe auf ihre Effizienz hin zu überprüfen und entsprechende Veränderungen zu realisieren.

2.2.1 Aufgaben des Usability-Engineering

Bei der benutzerorientierten Systemgestaltung lassen sich generell drei Aufgabenbereiche unterscheiden:

- *Anwendung und Umsetzung vorhandener Erkenntnisse*. In der Analyse bezieht sich die Anwendung und Umsetzung vorhandener Erkenntnisse vor allem auf Arbeitsprozesse, organisatorische Umgebung, Aufgaben- und Benutzerfähigketten-Analysen, Wissensmobilisierung von Nutzung, Schwachstellenanalyse. Bei der Gestaltung sind Kriterien und Richtlinien hinsichtlich der Dialoggestaltung, der Informationsgestaltung und der Benutzerführung gefragt. Die Evaluation kann Aussagen u. a. zu Aufgabenangemessenheit, Steuerbarkeit, Selbstbeschreibungsfähigkeit eines System machen.
- *Entwicklung und Einsatz von Methoden*. Es sollen Methoden bereitgestellt werden, die die Anbindung von benutzerspezifischen Anforderungen bereits in der Aufgabenanalyse und bei der Entwicklung eines benutzergerechten Designs ermöglichen. Dabei meint man die methodische Verknüpfung der benutzerorientierten Gestaltung mit den traditionellen Verfahren der Softwareentwicklung. Bei der Gestaltung sind Methoden zur Umsetzung der Gestaltungs-Richtlinien erforderlich. Die Evaluation erfordert Methoden zur Überprüfung und Bewertung der gestalterischen Massnahmen im Hinblick auf Aufgaben, Arbeitsprozesse und Benutzer.
- *Entwicklung und Verfeinerung von Werkzeugen*. Es werden Werkzeuge benötigt, die die Vorgehensweisen und Methoden zur Erzielung von z.B. Aufgabenangemessenheit bei der benutzerorientierten Gestaltung umsetzen. In der Analyse werden Werkzeuge zur Aufgabenanalyse benötigt, die die Charakterisierung der Aufgaben und der Benutzer unterstützen. Der Einsatz von Werkzeugen ist bei der Gestaltung erforderlich, wobei die Richtlinien und Standards die Rahmenbedingungen vorgeben. Aufgaben der Evaluation sind unter diesem Aspekt Prototyping, Qualitätssicherung, Bewertungskonzepte und unabhängige Prüfverfahren.

2.2.2 Das Nutzungskonzept

Die Erfahrung hat gezeigt, dass man den Begriff Nutzungskonzept explizit definieren muss, um im Vorfeld Missverständnisse auszuräumen. Wenn man im deutschen Duden nachschlägt, kann man die Vielfalt des Begriffes erahnen. Die Usability-Spezialisten haben bei der Nutzung des Wortes selbst mit Missverständnissen zu kämpfen. Je deutlicher man den Begriff

und seine Einbettung in seine Umgebung beschreibt, desto genauer kann man anhand dessen den Entwurfs/Entwicklungs-Prozess gestalten. Zur Definition des Nutzungskonzepts siehe auch Glossar.

USEWARE

Der neue Begriff USEWARE (**UW**) stammt ursprünglich von D. Zühlke [Zühl/Wahl'99]. Dieses neue Wort **Useware** steht für "interaktive Werkzeuge (SW und HW) und andere zusätzliche Mittel, die dem Benutzer zur Verfügung stehen " [Ober´99].

Die Suche nach der passenden Bedeutung des Begriffs UW erstreckt sich von System Image [Norm/Drap'86], Informationsvorrichtung [Norm'98], über Interaction Design [Wino'96&97], zu Informationsarchitektur [Coop'99], Computer im Kontext [Dahl/Math'93] und vieles mehr (u. a. [Laur'90], [Land'95] , [Niel'93]). Aber was bedeutet UW konkret?

Oberquelle unternahm den Versuch der Erklärung des Begriffs UW in seinem Manuskript: "Useware Design and Evolution: Bridging Social Thinking and SW Construction" [Ober´99].

Für die Grundanalyse unterscheidet er drei UW-Ebenen:
1. *Die Konzeptuelle Ebene*
2. *Die Interaktions-Ebene*
3. *Die Repräsentations-Ebene*

Die Konzeptuelle Ebene
Oberquelle versucht auf dieser Ebene dem Begriff Nutzungskonzept Nahrung zu geben. Entscheidend ist hier, dass die Eigenschaften des Nutzungskonzepts nicht auf sw-technischem Hintergrund basieren, sondern auf dem Kontext der Nutzung. Seine konzeptuelle Welt (wie er sie benennt) überdeckt sich mit der künstlichen Welt des Interagierens des Benutzers. Sie charakterisiert sich anhand von Beschreibung durch:
- räumliche Eigenschaften (z. B. Informationen über Bereiche, Mittel-Transfer).
- passive Mittel (z. B. Dokumentationsformen, Regelwerke).
- dynamische Aktivitäten (z. B. für einfache oder komplexe Manipulationsmechanismen).
- Agenten (z. B. semi-autonomous mechanisms, Internet-Agenten).
- andere Nachbarswelten oder Randgebiete (z. B. „interspaces“ [Wino'97]).
- Integration der virtuellen Welt in die physikalische Welt .

Einen anderen Ansatz der Abgrenzung des konzeptuellen Entwurfes von der technischen SW-Konstruktion liefert Züllighoven [Züll'98] in seinem WAM-Modell als einen Vorschlag zum benutzerorientierten Entwurf einer “vernetzten Welt” von Mittel, Werkzeugen und Mechanismen .

Die Interaktions-Ebene
Auf dieser Ebene sind die Mittel, mit deren Hilfe man mit der konzeptuellen Welt interagieren kann, von entscheidender Bedeutung. Sie müssen den Zugang zu Bereichen, anderen Mitteln, SW-Agenten, komplexen Manipulationsmechanismen, Nachbarswelten ermöglichen. Dem Benutzer muss es möglich sein, auf elementare Interaktionsfragen zur Orientierung, Aktion-Beeinflussung, Navigation, Ausweichmöglichkeiten Antwort zu finden. Ausserdem soll die Wahrnehmung des Benutzers automatisch unterstützt werden: z. B. bei dem Verstehen der konzeptuellen Welt („System-Help“), Sicherheitsaktionen („History“, „Undo“), Routineaktion-Behandlung (Definition von Macros).

Die Repräsentations-Ebene
Hier geht es um die Frage, wie sollen die konzeptuelle Welt und die Interaktionswelt repräsentiert werden. Gemeint ist nicht nur die Benutzeroberfläche sondern auch die systemtechnische Umgebung.

Die UW-Design Aktivität

Der Entwurf von Useware geschieht auf dem für Benutzer adäquaten Wege, um aus seiner Wahrnehmung die Schnittstelle zwischen Information und Interaktion optimal zu gestalten. Diese Aktivität ist ein kreativer und mentaler Prozess zwischen sozialen, aufgestellten Nutzungssituationen und formeller, künstlicher Konstruktion, mit dessen Hilfe man die Useware realisiert. Das heisst, **UW-Design** ist eine neue Herausforderung basierend auf einer Kombination aus sozialer Welt, technischem Know-how und Kommunikationsplattform.

Entscheidend ist also, die *Aktivitäten* der sozialen Welt in dem SW-Entwurf zu berücksichtigen. Oberquelle beschreibt diesen Vorgang so:
- *von* Anwendungen als einer Merkmalsammlung *zu* UW als vernetzte, konzeptuelle Welt, die integriert in die Benutzerpraxis ist.
- *von* der Schnittstelle als Übergang zwischen Kontext und technischem System *zur* konzeptuellen Welt (ihrer Abgrenzung, Handlung) angereichert mit sozialen Aspekten der Beobachtung, Kontrolle, des Einflusses.
- *von* Produkten mit festgelegtem Benutzerverhalten *zu* flexiblen Infrastrukturen für die Gewinnung von Feedback/Lernen über den Nutzungs-Kontext.
- *von* Benutzern, die die Lösungen des Engineers verstehen *zu* Engineers, die die UW verstehen.
- *von* der Anforderungs-Entwicklung als Vorbereitung der SW-Konstruktion *zu* UW-Design als Voraussetzung für den SW-Entwurf.
- *von* SW- Engineering *zu* „UW Design+SWConstruction+Evolution Support".

Fazit: Ein Useware-Designer muss zwei „Sprachen" beherrschen, nämlich die des Nutzungkontextes und der SW-Konstruktion. Im Kapitel 4 „Ein Prozess-Modell für UE" wird ein Versuch gemacht, einen gemeinsamen Formalismus zu entwickeln, um die o . g. „Sprachen" näher zusammenzuführen, um so die Integration von Software- und Usability-Engineering vollziehen zu können.

2.2.3 Erhebung von Szenarien

Ein **Szenario** ist eine episodische Beschreibung von Aufgaben und Tätigkeiten in ihrem Kontext [DAT-G´00], [Dzid/Frei00]. Unterschieden werden:

- das *Kontext-Szenario*, in dem die Aufgaben, die mit Hilfe der zu untersuchenden Software bearbeitet werden sollen, beschrieben werden. Das Kontext-Szenario enthält *keine Informationen über die tatsächliche Interaktion* mit der Software;

- das *Use-Szenario*, eine *episodische Beschreibung der Interaktion* mit der Software (z.B. ...Benutzer macht Doppelklick auf Schaltfläche x, Fenster Datei speichern erscheint, obwohl Fenster Datei öffnen erwartet wurde...), ohne dass für eine zu beschreibende Kern-

Aufgabe eine vollständige Abdeckung aller möglicher Interaktionen angestrebt wird. Use-Szenarien können ggf. durch "screen shots" erläutert werden oder ganz aus einem Videofilm bestehen, um die Objektivität der Beschreibung kritischer Nutzungssituationen zu sichern;

- der *Use-Case*, eine Kurzform des Use-Szenarios, der aus einer meist tabellarischen Gegenüberstellung von Benutzeraktionen und (Re-)Aktionen des Dialogsystems besteht.

Kontext-Szenario

Ein Kontext-Szenario dient der Erfassung des Nutzungs-Kontextes auf der Grundlage von ISO-9241-11. Die Erhebung von Kontext-Szenarien dient der Vorbereitung der Anforderungs-Entwicklung zu Beginn eines Entwurfs-Projektes. Es werden in der Sprache des Benutzers die Nutzungsbedingungen eines noch zu entwickelnden Systems beschrieben, um ein klares Verständnis der Arbeitserfordernisse zu erarbeiten. Die sachliche Richtigkeit und Vollständigkeit muss vom Benutzer bestätig werden (Validierung). In der nachfolgenden Tabelle (siehe Abbildung 2-1) sind die Leitfragen anzuwenden, um die Kontext-Szenarien zu erheben. Nach der Erhebung ist bei der Auswertung darauf zu achten, dass die Beschreibung von Daten des Nutzungs-Kontextes und die Ableitung von Anforderungen (Auswertungsschema) streng getrennt bleiben [DAT-G'00].

Abschnitt im Szenario	*Leitfragen (Erläuterungen hierzu im Auswertungsrahmen,Beispiele hierzu im Beispiel-Szenario)*
Einleitung	1. Formulieren Sie die Tätigkeit in einem oder in zwei Sätzen. 2. Aus welchen Aufgaben ist die Tätigkeit zusammengesetzt (typische Kern-Aufgaben aufführen, d.h., wenn grosser Zeitanteil oder häufig wiederkehrend oder sehr wichtig)? Welche dieser Kern-Aufgaben sollen durch die Software unterstützt werden? 3. Wie ist die Tätigkeit organisiert (z.B. als Mischarbeit, als Folge von Aufgaben, als monotone Einzelaufgabe)?
Voraussetzungen	4. Welche Qualifikation ist zur Bewältigung der Aufgaben erforderlich (Aufgabenbewältigung/ Softwarenutzung)? Welche Vorkenntnisse fehlen ggf.? 5. Wer (bzw. welches Ereignis) bestimmt, was zu tun ist? (Wer trifft die Auswahl? Selbständigkeit der Bearbeitung, Arbeitsteilung, externe Datenquellen) 6. Welche Hilfsmittel sind erforderlich (für die Aufgabenbewältigung/ zur Softwarenutzung)? Welche davon fehlen ggf., welche sind zusätzlich gewünscht?
Normale Durchführung	7. Welche Arbeitsschritte sind durchzuführen? 8. Welche Arbeitsschritte kehren häufig wieder? (Automatisierung gewünscht / erforderlich?) 9. Welche Arbeitsschritte werden automatisch durchgeführt? Sind bei automatisierten Arbeitsschritten Einflussmöglichkeiten des Benutzers vorhanden / erlaubt / gewünscht / erforderlich? 10. Kommt es vor, dass mehrere Benutzer gleichzeitig an dem gleichen Objekt (z.B. Vorgang, Akte, Dokument, Datensatz) arbeiten müssen? 11. Gibt es eine festgelegte Abfolge der Arbeitsschritte und wenn ja, wie sieht diese aus? (Ist eine Flexibilität sinnvoll / nötig?) 12. Welche Ergebnisse/ Teilergebnisse entstehen und wie werden diese ggf. verwertet / weitergeführt? 13. Welches Feedback bekommt die befragte Person in bezug auf die Arbeitsergebnisse und die Wirkung ihrer Arbeit?
Besonderheiten bei der Durchführung	14. Welche Unterbrechungen gibt es und warum? Welche Störungen treten auf (organisatorisch/ sozial/ technisch)? 15. Wie werden Fehler zurückgemeldet und behoben (organisatorisch/ sozial/ technisch)? 16. Welche wichtigen Sonderfälle müssen berücksichtigt werden (fallen dem Benutzer spontan ein; z.B. zur Arbeitsteilung/ Zusammenarbeit)?
Organisatorische Rahmen- bedingungen	17. Welche Organisationsziele gibt es im Hinblick auf die Tätigkeit? 18. Gibt es Mechanismen zur Leistungssteuerung/ Leistungskontrolle? (Wenn ja, welche? Sind diese erforderlich?) 19. Welchen Überblick hat Benutzer im Hinblick auf die Gesamttätigkeit? 20. Welche Änderungen, die die Aufgabenbearbeitung beeinflussen, sind zu erwarten oder werden gewünscht? Welche Vorschläge hat der/die Befragte dazu? 21. Von welchen Arbeitsergebnissen/ Arbeitsschritten sind Dritte (z.B. Kunden) direkt betroffen? Und was folgt daraus? 22. Welche Stressfaktoren gibt es und wie wird damit umgegangen?
Sonstige Bemerkungen zu bereits aufgetretenen Nutzungsproblemen.	Hier Beispiele sammeln, falls die befragte Person bereits während des Interviews etwas über aufgetretene Nutzungsprobleme berichtet. Im Allgemeinen werden solche Probleme besser anhand von Use-Szenarien analysiert.

Abbildung 2-1 Leitfragen zur Erhebung von Kontext-Szenarien [DAT-G'00]

Bei der Ableitung der Kontext-Anforderungen wird das Kontext-Szenario mittels eines Auswertungsschemas ausgewertet (siehe Abbildung 2-2).

Kontext-Szenario	**Erfordernisse aus der Aufgabe**	**Dialogprinzip**	**Anforderungen an den Dialog**	**Prüfkriterium**
…	…	…	…	…

Abbildung 2-2 Kontext-Szenario: Auswertungsschema [DAT-G'00]

Im folgenden werden *Auswertungs-Bausteine* (als Beispiel) zur Auswertung von Kontext-Szenarien vorgestellt [DAT-G´00], die dem Usability-Analytiker Erfahrungswerte bzgl. oft vorkommender Erfordernisse im Bürobereich, sowie damit verbundene Dialogprinzipien und oft vorkommende Anforderungen an das Dialogprinzip liefern sollen (siehe Abbildung 2-3):

Oft vorkommende Erfordernisse im Bürobereich	**Dialogprinzip**
- werden alle Arbeitsergebnisse der „key-task“ vollständig und korrekt erzielt? - ist die für die Aufgabenbearbeitung erforderliche Information unmittelbar verfügbar?	Aufgabenangemessenheit.
- ist der Stand der Bearbeitung erkennbar, so dass der Arbeitsperson die Tätigkeit klar ist? - kann die Person darüber entscheiden, wie die Arbeit fortzusetzen ist (es sei denn, die Folge der Schritte ist durch Prioritäten vorgegeben)?	Selbstbeschreibungsfähigkeit /Aufgabenangemessenheit. Steuerbarkeit.
- kann die Person darüber entscheiden, welche Aufgaben sie ausführen will (es sei denn, die Folge der Aufgaben ist durch Prioritäten vorgegeben)?	Steuerbarkeit, (Selbstbeschreibungsfähigkeit).
- sind alle Kenntnisse/Hilfsmittel gegeben, die für das Erledigen der Aufgabe erforderlich sind?	Selbstbeschreibungsfähigkeit /Aufgabenangemessenheit.

Abbildung 2-3 Oft vorkommender Erfordernisse im Bürobereich (Beispiel) [DAT-G'00]

Nachstehend werden einige typischen Anforderungen, die sich aus der Konkretisierung der Dialogprinzipien ergeben, aus dem Bürobereich vorgestellt (siehe Abbildung 2-4):

Aufgabenangemessenheit: -werden die aufgeführten „key task“ durch das Dialogsystem unterstützt? - wird ein häufiger Medienwechsel zw. Online-Infos und schriftlichen Unterlagen vermieden? - ist der Bearbeitungsstand so angezeigt, dass dem Benutzer die Bearbeitung des nächsten Arbeitsschrittes möglich ist? - bietet das System Hilfe zur fachlichen Aufgabenbearbeitung? - sind Prioritäten der Aufgabenbearbeitung angezeigt? - werden wiederkehrende Aufgaben besonders unterstützt? - sind Informationen, die zur Erledigung der Aufgabe notwendig sind , auf dem Bildschirm übersichtlich verfügbar? - weden Aufgaben, die das System selbst ausführen sollte, vom System automatisch ausgeführt?
Selbstbeschreibungsfähigkeit: - ist die Software mit den erworbenen Vorkenntnissen benutzbar? - gibt es ein kontext-sensitives Hilfesystem? - versteht der Benutzer das Gedankenmodell, das dem Programm zu Grunde liegt, um die Vorgänge bearbeiten zu können? - erkennt der Benutzer an den Systemrückmeldungen das Ergebnis seiner Arbeit? - wird dem Benutzer die erwartete Information angezeigt, um seine Aufgabe erledigen zu können?
Steuerbarkeit: - kann der Benutzer die nächste zu bearbeitende Aufgabe am System frei wählen? - ist dem Benutzer klar, wie der Dialog zu beginnen und fortzusetzen ist ? - kann der Benutzer den Dialog so steuern, wie es die Aufgabenteilung erfordert?

Abbildung 2-4 Oft vorkommende Anforderungen an das Dialogprinzip (Beispiel) [DAT-G'00]

Mit den oben vorgestellten Beschreibungs- und Auswertungs-Schemata kann man den Nutzungs-Kontext ermitteln und dokumentieren, anhand dessen man mit Hilfe von Use-Cases (für jede der Kern-Aufgaben des Nutzungs-Kontextes) die Use-Szenarien beschreiben kann, was im nächsten Abschnitt erläutert wird.

Use-Szenarien

Die Use-Szenarien dienen der Erfassung der Interaktion des Benutzers mit der Software in der gegebenen Nutzungssituation am Bildschirmarbeitsplatz. Es gibt zwei Situationen, in denen die Erfassung der Interaktion unterschiedlich sein kann [DAT-G'00]:

- In einem *Software-Entwicklungsprozess* (z.B. beim Prototyping) wird während der Anforderungs-Entwicklung die Interaktion des Benutzers für jede Kern-Aufgabe vollständig beschrieben, damit die aus dem Szenario abgeleiteten Anforderungen an das Dialogsystem mit den am Prototypen realisierten Merkmalen verglichen werden können. (Voraussetzung zur Entwicklung der Use-Szenarien ist die Erhebung des Nutzungs-Kontextes aus dem Kontext-Szenario.)

- In der Situation einer *Softwareprüfung* an einem existierenden System braucht die Interaktion nicht ausführlich erfasst zu werden; es reicht eine Erfassung in Form eines Use-

Case aus. Dialogschritte, die jedoch auf Nutzungsprobleme oder auf vermutete Normabweichungen hinweisen (critical incidents), sind ausführlich zu beschreiben und anschliessend auszuwerten (Normkonformitätsprüfung). Die kritischen Schritte sind als critical incidents hervorzuheben (z. B. Fettschrift).

Ein protokolliertes Use-Szenario ist während der Erhebung ein noch relativ ungeordnetes Dokument. Für den beschriebenen Tätigkeitsablauf muss eine (ggf. den Ablauf im Alltag idealisierende) *Ablaufstruktur der Aufgabenschritte* noch gefunden werden. Der interaktive Tätigkeitsablauf wird in einer zweispaltigen Tabelle als Use-Case beschrieben (siehe Abbildung 2-5). Eine weitere Tabellenspalte enthält zu den kritischen Dialogschritten (critical incidents) die zugehörigen Anforderungen, gegen die vermutlich verstossen wird, oder die zu klärenden (Prüf-)Fragen. Eine Illustration kritischer Nutzungssituationen mittels "screen shots" ist nützlich, um die beschriebenen Nutzungsprobleme besser zu veranschaulichen.

Aufgabe in Teilaufgaben zerlegen	***Use-Case***		**Anforderungen oder zu klärende Fragen**
	Benutzer-Aktion	**Aktion/Reaktion des Dialogsystems**	
...	...	...	...

Abbildung 2-5 Use-Szenario: Auswertungsschema [DAT-G'00]

Es ist hier ebenfalls, wie bei Kontext-Szenarien, darauf zu achten, dass die Beschreibung und Bewertung streng getrennt bleiben.

Beispiel für ein Use-Szenario anhand von WinZip [DAT-G´00]

Herr Müller ist ein fleissiger wissenschaftlicher Mitarbeiter. Er will die sehr umfangreiche Datei "literaturliste" auf Diskette speichern. Die Datei "literaturliste.doc", die er seinem Chef, Herrn Schmidt, geben will, passt nicht auf eine Diskette. Die Datei ist nämlich 1,9 MB gross. Herr Müller nutzt gelegentlich das Programm WinZip zum Komprimieren von Dateien, das auf jedem Rechner zur Verfügung steht (siehe Abbildung 2-6).

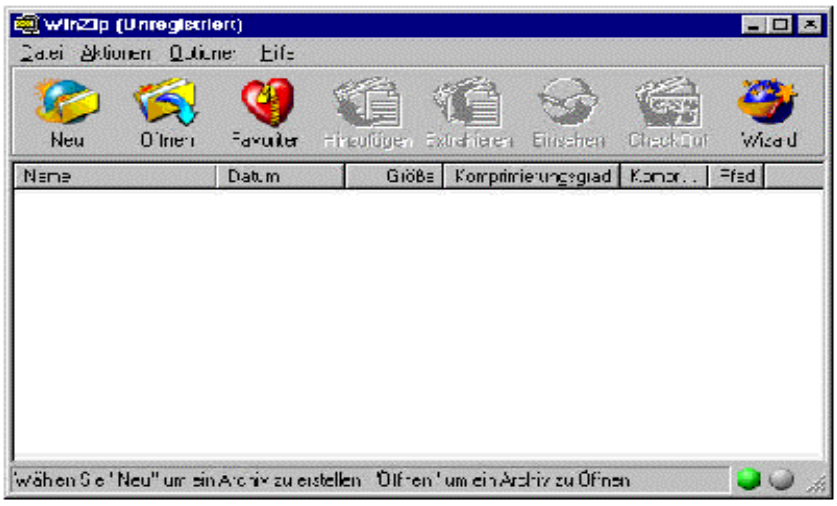

Abbildung 2-6 Programm WinZip [DAT-G´00]

Analyse des Nutzungsproblems [DAT-G'00]:

Der Benutzer geht mit Hilfe seines Nutzungs-Konzepts, nämlich eine Datei auszuwählen, die komprimiert werden soll, in den Dialog mit dem Programm und gerät in eine Sackgasse: der Designer ist beim Entwurf der Benutzungsschnittstelle und insbesondere bei der Benutzerführung offensichtlich davon ausgegangen, dass dem Benutzer die Idee des Zip-Werkzeugs bekannt ist. Zip ist ein Komprimierungswerkzeug für einzelne Dateien und komplexe Verzeichnis/Unterverzeichnis-Strukturen und existiert für die meisten Plattformen. Ursprünglich kommt diese Idee aus der UNIX-Welt, die allerdings in WinZip nicht zu Ende gedacht wurde. Es ist eigentlich ein zusammengesetztes Werkzeug aus den beiden UNIX-Befehlen "compress" (komprimiert Dateien) und "tar" (tape archiver). Aus dem Werkzeug tar stammt das Konzept der Archive, da "tar" insbesondere dazu benutzt wird, Dateibäume (Verzeichnisse/ Unterverzeichnisse) platzsparend auf Bändern oder Platten zu archivieren, wobei die Baum-Struktur unveränderbar bleibt (übrigens in UNIX beim Komprimieren die neusten tar-Versionen unterstützen automatisch compress, gzip).

Normalerweise benötigt man aus Benutzersicht das Konzept Archiv nicht, wenn man nur eine einzelne Datei komprimieren möchte. Dieser Spezialfall sollte für den Benutzer so gestaltet sein, dass er entweder im Dialog mit dem Programm in das Konzept des Archivs eingeführt wird, oder es aber nicht kennen muss, um seine Aufgabe zu erfüllen. Benutzer, die nur gelegentlich Dateien verschicken und selten in die Situation kommen, sehr grosse Dateien kompakter zu machen, um sie versenden zu können, haben nur die Aufgabe "Datei komprimieren" vor Augen und brauchen in dieser Situation nicht zu wissen, dass sie auch mehrere Dateien verschicken können, deren Verzeichnisstruktur erhalten bleiben soll. Diese Situation hat der Designer jedoch weder beim Entwurf der Benutzungsoberfläche noch bei der Benutzerführung (in den Hilfetexten) berücksichtigt. Ein Einarbeitungsproblem wie das oben beschriebene wäre während des Prototyping des Programms aufgetreten und somit aufgedeckt worden. Prototyping mit verschiedenen Benutzern hätte dem Designer ermöglicht, eine an-

dere Sichtweise auf die Arbeitsaufgabe einzunehmen und die hier beschriebene Nutzungssituation kennen zu lernen.

Unter Beibehaltung des Systemkonzepts "Archiv" hätte der Designer die Chance gehabt, die Rückmeldungen des Systems auf die Nutzungssituationen anzupassen. An typischen Einarbeitungsproblemen wie dem hier analysierten, kann der Prüfer sofort erkennen, dass die Designer den Usability-Prototyping-Prozess ausgelassen haben.

Das Nutzungsproblem beruht vermutlich auf der Unvereinbarkeit des Nutzungskonzepts des Benutzers und des Systemkonzepts, das der Entwicklung des Programms zugrunde lag.

Vermutete Normabweichung [DAT-G'00]:

Der Dialogablauf folgt vermutlich nicht dem erwarteten aufgabenbezogenen Ablauf des Benutzers (Nutzungs-Konzept), vielmehr wird vom Benutzer verlangt, einem ihm möglicherweise nicht vertrauten Systemkonzept zu folgen (hier die Konvention "Archiv").

⇒ ISO 9241-10: "Die Form der Eingabe und Ausgabe sollte der jeweiligen Arbeitsaufgabe und den Benutzerbelangen angepasst sein."

- Die Meldungen des Systems und die angebotene Hilfe scheinen unverständlich zu sein.

  ⇒ ISO 9241-10: "Rückmeldungen oder Erläuterungen sollten genau auf die Situation bezogen sein, für die sie gebraucht werden, um ihren Wert für den Benutzer zu erhöhen."

Nachfolgend eine User-Analyse anhand des WinZip User-Szenarios [DAT-G'00]:

Die Durchführung der Aufgabe wird hier anhand des Dialoges zwischen dem Benutzer und dem System dokumentiert und ausgewertet (siehe Abbildung 2-7).

Aufgabe in Teilaufgaben zerlegen	***Use-Case***		**Anforderungen oder zu klärende Fragen**
	Benutzer-Aktion	**Aktion/Reaktion des Dialogsystems**	
Teil-Aufgabe 1*:* **zu komprimierende Datei auswählen**	M. klickt Icon „Neu“ an.	Das Programm öffnet ein neues Dialogfenster. **critical incident: Das Dialogfenster ist nicht zur Auswahl der zu komprimierenden Datei vorgesehen**	Das neue geöffnete Dialogfenster muss die Auswahl der zu komprimierenden Datei ermöglichen. ISO 9241-10 „Die Form der Eingabe und Ausgabe sollte der jeweiligen Arbeitsaufgabe und den Benutzerbelangen angepasst sein.

	M. sucht nach Verzeichnis „Liste„ um dort die Datei „literaturliste.doc„ auszuwählen.	Das Programm lässt das Auswählen von Verzeichnissen und Dateien zu **Critical incident: M. glaubt immer noch, dass er die zu komprimierende Datei auswählen kann**.	
	M. sieht zwar das Verzeichnis „Liste“ aber die Datei„literatutliste.doc ist nicht vorhanden. Jetzt sieht M., dass bei „Dateityp“ der Dateityp „zip-files“ ausgewählt ist.		
	M. versucht jetzt „Alle Dateien“ auszuwählen	Man kann eine Auswahl treffen und der Eintrag „Dateityp“ wird als ausgewählt angezeigt. **Critical incident: M. bemerkt nicht, dass das Programm diese Option nur anbietet um komprimierte Dateien mit beliebigen Dateiendungen anzuzeigen.**	
	M. klickt auf OK, um dem Programm mitzuteilen „das ist die Datei die ich komprimieren will„	Ein Meldungsfenster erscheint: „..kann nicht erzeugt werden...„. **Critical incident: M. versteht die Meldung nicht.**	ISO 9241-10 „Rückmeldungen oder Erläuterungen sollten genau auf die Situation bezogen sein, für die sie gebraucht werden, um ihren Wert für den Benutzer zu erhöhen.„.
	M. klickt auf „ja„ , da diese Icon defaultmässig ausgewählt ist.	Ein Meldungsfenster erscheint: „Datei kann nicht geöffnet werden...„ **Critical incident: M. versteht die Meldung nicht.**	ISO 9241-10 „Rückmeldungen oder Erläuterungen sollten genau auf die Situation bezogen sein, für die sie gebraucht werden, um ihren Wert für den Benutzer zu erhöhen.„
	Herr M. klickt auf OK	Das Programm zeigt wieder das Programmfenster.	
	Jetzt geht M. ein Licht auf! Er versteht, dass er im Moment der später zu komprimierenden Datei einen Namen geben soll. M. tippt folgenden Dateinamen ein:„literaturliste.zip"„	Programm öffnet neues Fenster, in dem die zu komprimierende Datei ausgewählt werden kann.	

Teilaufgabe 2: **Datei komprimieren**	M. klickt die Datei literaturliste.doc an und klickt auf „Hinzufügen„.	Das Programm führt jetzt die Komprimierung durch. Dies dauert einige Sekunden und wird durch eine Fortschrittsanzeige signalisiert. Anschliessend erscheint das Programmfenster wieder.	
	M. will jetzt wissen, wie gross die komprimierte Datei ist. In der Liste vermutet er unter „Kompri...“ zu erkennen, dass die Datei 0,299685 Mbyte gross ist.		
Teilaufgabe 3: **Datei auf Diskette kopieren**	M. startet das Programm „Explorer“ um die Datei auf Diskette zu kopieren. Dies funktioniert problemlos. M. hofft jetzt, dass sein Chef die Datei lesen kann, da sie nicht im Wordformat ist.		

Abbildung 2-7 Auswertungsschema für WinZip Use-Szenarien[DAT-G'00]

2.2.4 Die Entwicklung der Anforderungs-Definition

Im Folgenden wird die Anforderungs-Definition als das Ergebnis des Usability-Prozesses betrachtet und ihre Entwicklung als Prozess einer Konsensbildung. Der Prozess hat folgende Aufgaben: Missverständnisse zwischen Beteiligten klären, Lösungsvorschläge besprechen, Kompromisse suchen – aber nicht auf Kosten der Gebrauchstauglichkeit, Mindestqualität der Anforderungen prüfen auf Effektivität und Effizienz und Zufriedenstellung.

Validierung von Analyse und Entwurf

Dzida/Freitag sehen das Ergebnis der Anforderungs-Entwicklung als einen Lösungsvorschlag für den Entwurf von SWP [Dzid/Frei´98]. In der Abbildung 2-8 werden die Voraussetzungen, die zu einem Entwurfsvorschlag führen, dargestellt.

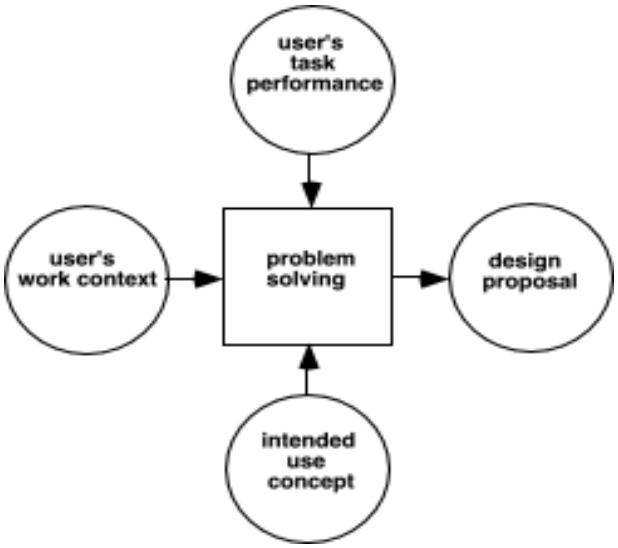

Abbildung 2-8 Anforderungs-Entwicklung als Problemlösungs-Prozess [Dzid/Frei´98]

Die Inputs sind der Arbeitskontext des Benutzers, die Aufgaben des Benutzers und das intendierte Nutzungskonzept [Broo´95]. Die Forderung nach einem aktiven Informationsaustausch zwischen dem Benutzer und dem Analytiker ist eine der wichtigsten Voraussetzungen bei der Anforderungs-Entwicklung [Goug´96]. Denn in der Praxis hat man oft beobachten können, dass der Benutzer nur selten allein sein Nutzungskonzept genaustes definieren kann.

Um den Problemlösungs-Prozess zu erklären, muss man sich klar darüber werden, welche unterstützenden Massnahmen zum Ergebnis beitragen. Dzida/Freitag sind der Meinung, dass der dialektische Prozess dem Synthese-Prozess vorausgehen muss, um einen Lösungsvorschlag erzeugen zu können (siehe Abbildung 2-9). Im Kapitel 4 werden die beiden Prozesse genauer vorgestellt. Hier sei nur darauf hingewiesen, dass es sich um Begriffe handelt, die aus dem „dörnerschen Problemlösen“ [Dörn´79] der dialektischen und Synthese-Barriere stammen (Psychologie des Problemlösens). Diese theoretischen Konstrukte kann man auf das Problemlösen bei der Anforderungs-Entwicklung anwenden, um Widersprüche, Missverständnisse aufzudecken bevor man zum SW-Entwurf der SW-Entwicklung übergehen kann.

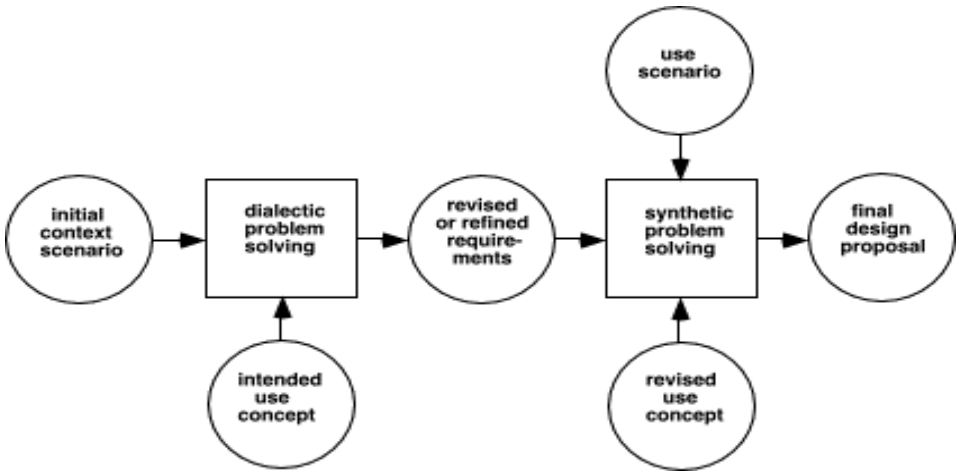

Abbildung 2-9Zwei Problemlösungstypen [Dzid/Frei´98]

Das Ergebnis des Synthese-Prozesses ist ein validierter Lösungsvorschlag; zugleich aber ist es notwendig zu bedenken, dass unter bestimmten Umständen einzelne Anforderungen nicht vollständig verstanden werden könnten, und weil sie einem interaktiven Ablauf unterliegen, der erst durch Nutzungserfahrung geklärt werden kann und ggf. nachgebessert werden muss [Dahl/Math'95]. Deswegen wird zusätzlich eine **Claims Analysis** [Carr/Ross´92] an dieses Prozess-Modell angeschlossen, um die Vor- und Nachteile von Design-Entscheidungen zu dokumentieren und abzuwägen (siehe Abbildung 2-10). Auf diese Weise alternieren die Prozesse der dialektischen Aufklärung von Widersprüchen und der (Synthese-) Konstruktion von Lösungen. Der Synthese-Prozess also induziert einen Lösungsvorschlag und in vielen Fällen ein sog. *innovatives Kontext-Szenario*, das als möglicher Nebeneffekt zur Optimierung der zukünftigen Geschäftsprozesse oder Inszenierung neuer Geschäftsideen hervorgeht.

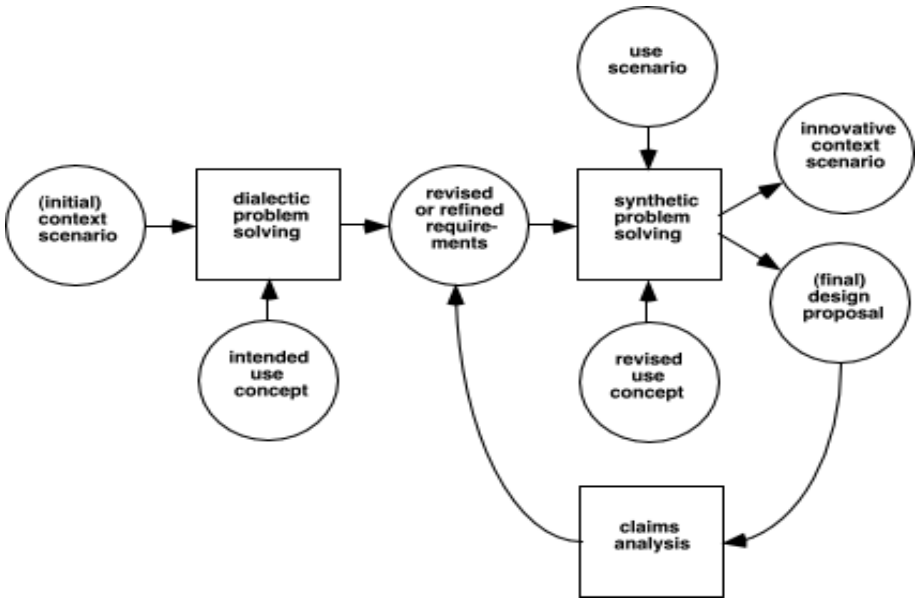

Abbildung 2-10 Iteratives Prozess für Anforderungs-Entwicklung [Dzid/Frei´98]

Der *Claims Analysi*s liegt eine Formel zugrunde, die besagt, dass eine Technologie zugleich wünschenswerte aber auch nicht wünschenswerte Konsequenzen in sich birgt. Das heisst, um zu entscheiden, was überwiegt, muss man sich in eine *„Pro- und Contra-Aufstellung“* begeben, die von einem bestimmten Kontext abhängig ist. Denn nur durch solche Entscheidung kommt man zu einem zufriedenstellenden Ergebnis. An einem einfachen Beispiel kann man sich das verdeutlichen: wenn man Icons kontextunabhängig betrachtet, sind sie in erster Linie sehr hilfreich (also ein Pro), um sich schnell zurecht zu finden. Andererseits, wenn man sie kontextabhängig betrachtet, können sie die „falschen“ sein oder zu viele auf einer Web-Seite sein, so dass man sich nicht zurecht finden kann (also ein Contra). Die *Claims Analysi*s ist also eine Massnahme, die man ergreifen muss, um im Kontext Entscheidungen darüber zu treffen, welche Merkmale des zukünftigen SWP einen Nutzen oder keinen Nutzen versprechen, oder noch nicht verstanden worden sind.

In ihrer Veröffentlichung betonen Dzida/Freitag die Wichtigkeit der Erhebung von Szenarien. Denn der Versuch des Verstehens - seitens des Herstellers der Benutzeraufgaben anhand von Szenarien, und seitens der Benutzer software-technischen Möglichkeiten - ist der Schlüssel zur *gemeinsamen* Anforderungs-Entwicklung. Eine valide Anforderungs-Definition wird von den Autoren als eine auf Konsens beruhende Beschreibung angesehen.

Die Erstellung eines Brückenschlags zwischen Software- und Usability-Engineering betrachten Dzida/Freitag als eine zentrale Aufgabe zukünftiger SW-Entwicklungsprojekte (siehe Abbildung 2-11). Um dies zu erreichen, müssen das Usability-Prototyping sowie die frühzeitige Validierung von Anforderungen und Lösungsvorschlägen professionalisiert werden.

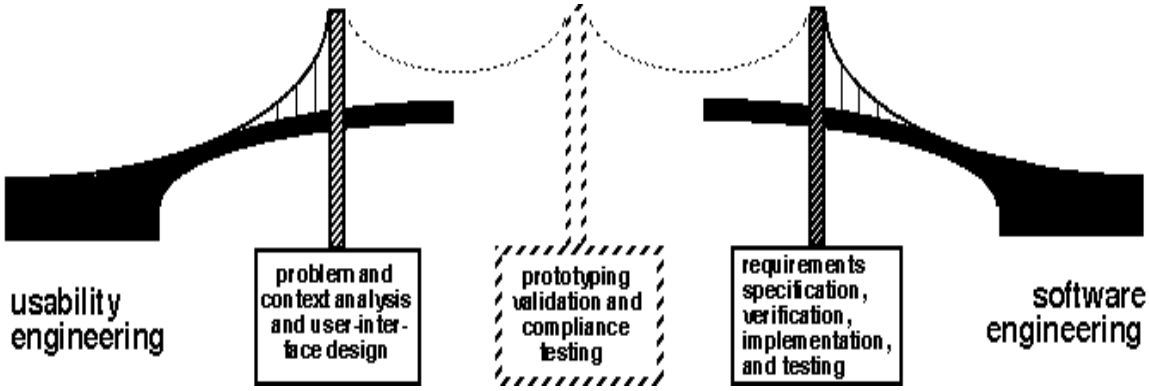

Abbildung 2-11 Brücke zwischen Usability- und Software-Engineering [Dzid/Frei´98]

Prototyping

SW-Prototyping bedeutet nicht zwingend Usability-Prototyping – das soll hier in diesem Unterkapitel u. a. deutlich gemacht werden:

Es gibt in der Tat unterschiedliche Prototyping-Ansätze und -Methoden. Sie unterscheiden sich nicht nur durch ihr Vorgehen sondern auch durch ihre Zielsetzung, d.h. was man dadurch erreichen will [Floy et.al´92]. Es gibt folgende ***Prototyping-Arten***:

- *Piloting* (technisches Prototyping), um die technische Machbarkeit zu demonstrieren, oder eine neue Technologie zu veranschaulichen, ohne dabei schon eine konkrete Anwendung im Blick zu haben (Abbildung 2-12).
- *Experimentelles Prototyping* (technisches Prototyping) [Bisch/Pomb´92], [Hsia et.al´93], um die Machbarkeit (Optimierung, Schnelligkeit, Integrationsgrad) zu testen. Dazu gehören u. a. SW-Prototyping, funktionales Prototyping, „Throw-Away-Prototyping“ (Wegwerf-Prototyping, um Erfahrungen zu sammeln).
- *Evolutionäres Prototyping* (Usability-/ technisches Prototyping), um die Anpassung des Prototypen an sich ständig verändernde Anforderungen aufzuzeigen. Dazu gehören u. a. inkrementelles Prototyping, analytisches Prototyping. Ein Nachteil des Prototypings sind die hohen Durchführungskosten.
- *Exploratives Prototyping* (Usability-Prototyping), um das gemeinsame Verständnis eines SWP und seiner Einsatzbedingungen zu erarbeiten; realisierbar u. a. durch User-Interface-Prototyping („Look&Feel“, Oberflächensichtung), „Rapid-Prototyping“ (schnelles Anschauungsobjekt), konzeptionelles Prototyping, T-Modell-Prototyping (Abbildung 2-13).

Demnach gibt es auch unterschiedliche ***Prototyp-Arten***, die unterschiedliche Schwerpunkte haben können:

- *Low-Fidelity* sorgt für eine anschauliche Diskussionsbasis mit geringen Interaktions- und Funktionsfähigkeiten [Isen et.al '96];
- *High-Fdelity* hat ausgearbeitete Funktionalität und ausführbare Aktionen, z.B. Grafik-Anwendung, ein einzelner Roboter-Prototyp [Isen et.al '96];
- *Pilot-System* zeigt die neuen technologischen Möglichkeiten auf, z. B. ein Forschungs-SWP, Produkte zum Ausprobieren;
- *Horizontaler Prototyp* dient meistens der Demonstration der gesamten Benutzungsoberfläche, z. B. erstellt mit Hilfe von SW-Oberflächenbausteinen wie GUI, Style-Guides, Menü-Bausteine [Budd et.al.'92];
- *Vertikaler Prototyp* dient der Erprobung der interaktiven Funktionalität ausgewählter Teilbereiche eines SWP (in die „Funktionale Tiefe gehen“), z.B. mit Hilfe der Simulation einzelner Dialogsequenzen [Budd et.al.'92].
- *T-Modell-Prototyp* ist eine Kombination aus einem horizontalen Prototyp und vertikalen Prototyp – als Ergebnis des explorativen Prototypings.

Die Unterschiedlichkeit des Usability-Prototyping und ihre Abgrenzung gegenüber den technischen Prototyping-Methoden (Piloting) und Prototyp-Arten zeigen Dzida/Freitag in ihrer Veröffentlichung auf. Usability-Prototyping ist eine Methode, die zur Validierung und zum Testen von Entwurfslösungen dient. Sie nehmen eine deutliche Teilung vor, die in der Abbildung 2-12 dargestellt ist.

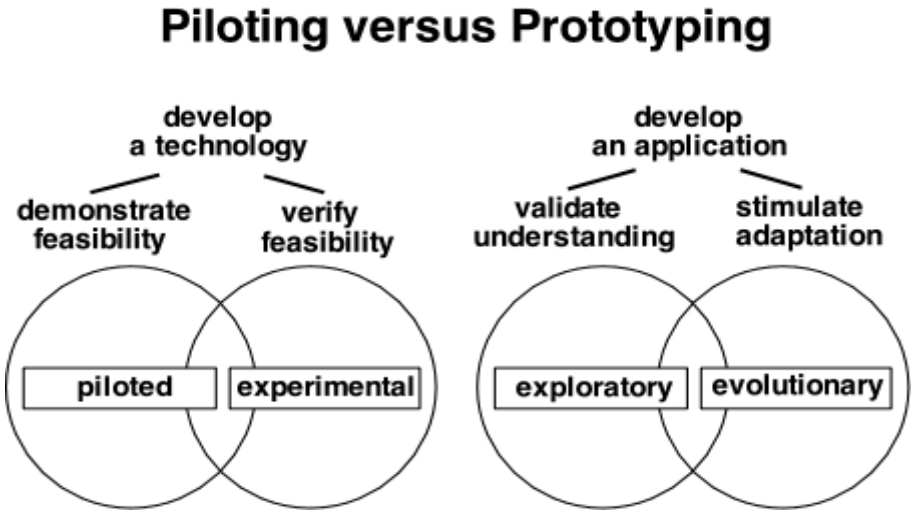

Abbildung 2-12 Prototyping-Arten [Dzid/Frei´98]

Beim technischen Prototyping kommt es vor allem auf software-technische Machbarkeitskriterien an, wie Integrationsgrad, Schnelligkeit, Korrektheit; beim U-Prototyping allerdings auf Verständlichkeit, Gebrauchstauglichkeit. Die entscheidende Schlussfolgerung hier lautet: *was nutzt uns ein korrektes SWP, wenn wir es nicht gebrauchen können, weil es den Zweck verfehlt!*

U-Prototyping verursacht zwar zu Beginn der Entwicklung einen Aufwand, spart aber Kosten für jeden während des Prototyping aufgedeckten Fehler/Mangel – nach Einführung des SWP am Arbeitsplatz liegen die Kosten für die Beseitigung des Mangels etwa sechzigmal höher [Bias/Mayh'94]. Werden ungünstige Arbeitsabläufe, die die Effektivität und Effizienz der Arbeit verringern und zu Unzufriedenheit bei den Benutzern führen, erst entdeckt, nachdem das SWP am Arbeitsplatz installiert wurde, entstehen zusätzliche hohe Kosten durch den erhöhten Nutzungsaufwand.

Das explorative Prototyping wird als Entwurfsmethode eingesetzt, um die Anforderungs-Definition in einem Team von Benutzern, Entwicklern, etc. zu erarbeiten. Ein Prototyp dient der Veranschaulichung der Anforderungs-Definition [Dzid/Frei´00]. Ein Prototyp ist die Darstellung einer Entwurfs-Idee, d.h. die Umsetzung von Nutzungs-Anforderungen des Benutzer in Produkt-Attribute/Merkmale [Hill/Houd´97]. Es bietet dem interdisziplinären Prototyping-Team die Möglichkeit ‚Anforderungen zu konkretisieren, zu prüfen, Entwurfsprobleme aufzudecken, Design-Alternativen abzuwägen. Der Prototyp zeigt auf , ob und inwieweit die Anforderungen, die Benutzer und Organisation an die SW stellen, verstanden und erfüllt wurden.

Der Einsatz von Entwicklungs- und Prototypingwerkzeugen beim Entwurf von Benutzerschnittstellen garantiert noch keine Qualität an sich. Werkzeuge sind nur Hilfsmitteln im Prototyping-Prozess. Die Qualität des Produktes hängt nicht nur davon ab, ob Prototyping statt-

gefunden hat, sondern auch davon *Wie* diese Prozesse organisiert sind. Eine „Kultur des Ausprobierens“ [Dzid/Frei´00] sollte gefördert werden. Was beim Prototyping gefragt ist, ist die Bereitschaft, neue Arbeitsmöglichkeiten auszuprobieren; konstruktiv und kooperativ sollte die Tauglichkeit neuer Nutzungsmöglichkeiten an der SW erprobt werden, insbesondere dann, wenn sie dem Benutzer und Hersteller nicht ganz klar sind.
Hier ist ein Umdenken bei allen Beteiligten gefordert. Es ist möglich, ein Kriterium für das Terminieren des Prototyping-Prozesses anzugeben, nämlich der Nachweis der vom Benutzer akzeptierten Mindestqualität an die Nutzungs-Eigenschaften des SWP.

T-Modell-Prototyping

Das Usability-Prototyping ist im Wesentlichen das Explorative Prototyping. Das Ergebnis Realisierung des Explorativen Prototypings ist das von Nielsen vorgestellte „T-Modell“ (siehe Abbildung 2-13) [Niel´93].

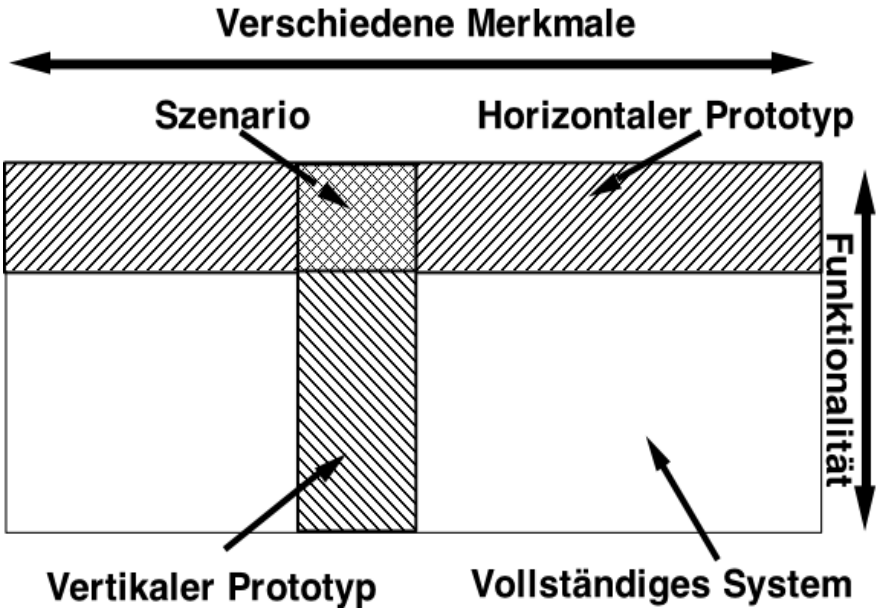

Abbildung 2-13 T-Modell-Prototyping [Niel´93]

Unter einem Prototyp versteht Nielsen Nutzungs-Szenarien, die auszuprobieren sind. *Horizontale Prototypen* umfassen möglichst alle Oberflächen-Merkmale des späteren Systems, zeigen aber keine durchgängige, interaktive Funktionalität. Bei *Vertikalen Prototypen* dagegen wird nur ein zentraler Teil der Systemfunktionen berücksichtigt. Dabei jedoch soll der Prototyp weitgehend (für die ausgewählten Merkmale) bereits vollständige Interaktion (Funktionalität) deutlich machen können. Szenarien sind bei Nielsen sozusagen abgespeckte Prototypen, die die Schnittmenge von horizontalen und vertikalen Prototypen abdecken. Den grossen Vorteil in solchen Prototypen sieht Nielsen in der Schnelligkeit, in der sie erstellt und auch verändert werden können. Hierfür muss nicht unbedingt ein Programmieraufwand entstehen, denn je nach Umständen kann ein Papier-Bleistift-Entwurf ausreichen.

Der **Prototyping-Prozess** besteht aus drei Phasen: Vorbereitung, Durchführung und Bewertung – veranschaulicht in Abbildung 2-14

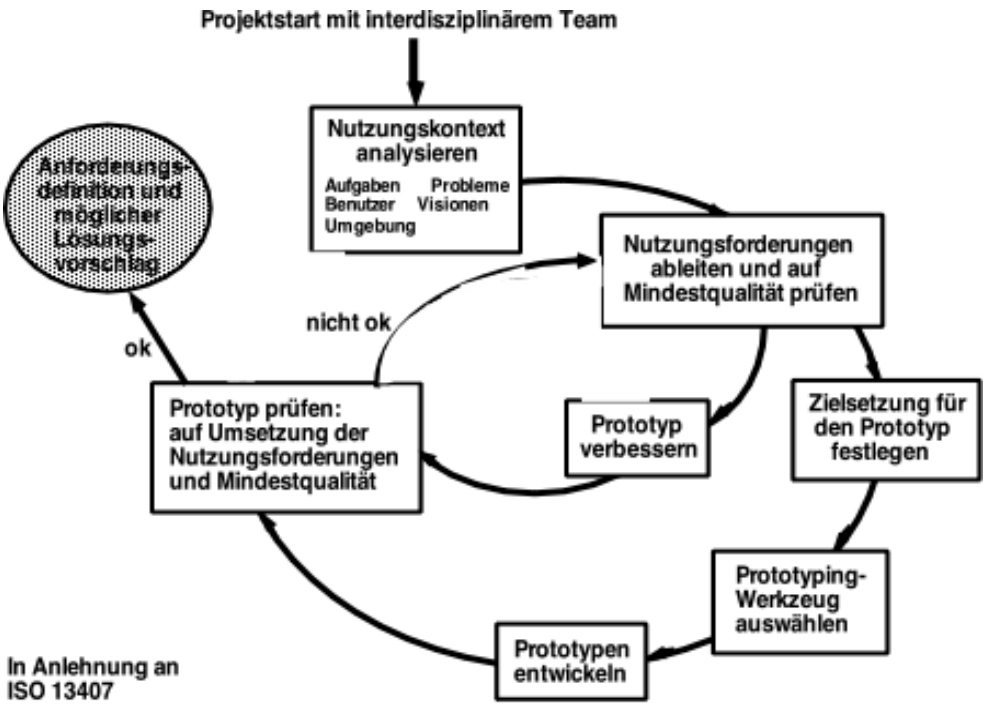

Abbildung 2-14 Prototyping-Prozess in Anlehnung an ISO 13407[Dzid/Frei´00]

Ein Ablaufbeispiel kann folgendermassen aussehen:
(1) erarbeite im Team Nutzungsforderungen anhand des Nutzungs-Kontextes – konkretisiere und ergänze sie,
(2) prüfe die merkmalsbezogenen Anforderungen auf Mindestqualität (ISO 9241-10, bis17) und leite weitere Merkmalsanforderungen aus dem Nutzungs-Kontext ab,
(3) entwirf einen Prototypen,
(4) prüfe den Prototypen im Team auf Mängel – falls Mängel existieren, dann zurück zu (1) oder (3),
(5) dokumentiere die Anforderungs-Definiton und den darin enthaltenen Lösungsvorschlag.

Der Prototyping-Prozess hat drei wesentliche *Eigenschaften*:

- er hilft bei dem Verstehen von unklaren Tätigkeits-Anforderungen und schwer vorstellbaren Produkt-Merkmalen gleichzeitig [Jark et.al'98];
- liefert eine doppelte Darstellung des künftigen SWP (in Form von Tätigkeitsbeschreibungen; Use Szenarien und vorgeschlagenen Produkt-Merkmalen; Prototyps);
- hilft Einarbeitungsprobleme zu erkennen und beseitigen.

Die folgenden **Grundsätze bei der Durchführung des Prototypings** und Hinweise für den Benutzer dienen der besseren Kommunikation und Erfüllung des Zwecks des Vorhabens während des Prozesses:

- die Ausgestaltung des Prototyps hängt von dem zu lösenden Nutzungsproblem ab;
- die Teilnehmer des Prototypings müssen auf die Präsentation eines Prototyps vorbereitet sein, denn Prototypen veranschaulichen nur ausgewählte Aspekte des zukünftigen Systems und den Beteiligten muss klar sein, wie der Änderungs-Prozess abläuft (transparentes Change-Prototyping);
- ein Prototyp kann verschiedene Formen haben: *Prototyping-Wekzeuge* reichen von „Papier und Bleistift“ über Präsentationsgrafik, GUI-Builder, Internet bis hin zu komplexen Simulationsgeräten;
- der Zeitpunkt der Präsentation des ersten Prototyps muss nach bestimmten Gesichtspunkten (wie Prototyp-Reife, -Verständnis, kein Perfektionismus ist gefragt) abgewogen werden;
- spätere Prototypen sollen dem Benutzer die interaktive Aufgabenausführung am System erlauben;
- Entscheidungen im Prototyping-Prozess müssen auf Konsensbildung beruhen;
- der Benutzer soll einen Prototypen immer danach beurteilen, welche Verbesserung er für die Tätigkeit bietet, und sich nicht von „tollem“ Bildschirmlayout beeindrucken lassen;
- wenn noch Fragen offen stehen, soll der Benutzer nach Design-Alternativen verlangen;
- der Benutzer soll sich während der Prototypenentwicklung über den Stand der Umsetzung von Anforderungen informieren lassen;
- das Ziel des Ausprobierens ist immer unter dem folgenden Aspekt zu verfolgen: die Umsetzung von Tätigkeits-Anforderungen in technische Merkmale des SWP.

Die Frage, die sich als letzte stellt ist, wie geht es nach dem Prototyping weiter? Nun, die Anforderungs-Definition ist ein (vorläufiges) Ergebnis des Prototyping-Prozesses und Grundlage des SW-Entwurfs. Zu bemerken ist aber, dass bei der realen Nutzung des SWP Nutzungs-Probleme entstehen können, oder sich neue Anforderungen ergeben können. Die Anforderungs-Definition unterliegt somit einem Nachbesserungsprozess (Pflege- Prozess), in dem sie später nachgebessert, aktualisiert wird.

Zusammenfassend kann man festhalten, dass ohne das explorative Prototyping keine gebrauchstauglichen SWP produziert werden können; ein Prototyp die technischen Möglichkeiten aufzeigt, um Nutzungsforderungen an das SWP zu erfüllen; Design-Alternativen bei der Konsensbildung helfen; Prototypen und Szenarien die Basis der Anforderungs-Entwicklung sind; Use-Szenarien die Umsetzung von Tätigkeits-Anforderungen in Produkt-Merkmale (Prototyp) unterstützen; Prototyping als eine „Kultur des Ausprobierens“ [Dzid/Frei´00] in den Entwicklungs-Prozessen selbstverständlich werden muss.

Anforderungs-Definition

Als Grundlage der Definition der *Anforderungs-Definition* dient der Begriff Anforderung [ISO-8402]: „Expression of the needs and/or their translation into stated requirements for the characteristics of an entity (par.2.3)“.

Die Anforderungs-Definition ist demgemäss eine Gegenüberstellung von *Tätigkeits-Anforderungen* und *Produkt-Merkmalen*. Sie besteht aus einer Tabelle (siehe Abb. 2-15 mit kleinem Beispiel), die in einer Spalte die Anforderungen des **Lasten-Heftes** enthält und in der anderen die korrespondierenden Anforderungen des **Pflichten-Heftes**. Das Lasten-Heft liefert dabei die Begründung für das Pflichten-Heft, und zwar Zeile für Zeile. Falls es keine Begründung im Lasten-Heft gibt, dann ist das entsprechende Produkt-Merkmal des Pflichten-Heftes unbrauchbar.

Lasten-Heft: *Tätigkeits-Anforderungen*	**Pflichten-Heft:** *Produkt-Merkmale*
▪ der Benutzer soll die augenblickliche Aufgabe unterbrechen können, weil er zwischendurch telefonieren muss. ▪ ... ▪ usw.	▪ der Dialog muss unterbrechbar sein und ohne Datenverlust wieder aufgenommen werden können. ▪ ... ▪ usw.

Abbildung 2-15 Struktur der Anforderungs-Definition [Dzid/Frei'00]

Beispiel:
Erfordernis des Nutzungs-Kontextes soll hier lauten: von aussen eingehende Telefonate müssen beantwortet werden können - dies ist ohne Widerspruch zu akzeptieren, weil es zum Selbstverständnis einer Dienstleistung gehört;

Aufgaben-Erfordernis dagegen: komplizierte Aktenlage muss störungsfrei bearbeitet werden können.

Die Anforderungen zusammengefasst in einer Tabelle (siehe Abbildung 2-16):

Lasten-Heft: Tätigkeits-Anforderungen	**Pflichten-Heft:** Produkt-Merkmale
▪ bei der Bearbeitung einer komplizierten Aktenlage soll der Benutzer durch Telefonate nicht gestört werden. ▪ eingehende Telefonate müssen stets beantwortet werden können.	▪ Umleitung der Telefonate. ▪ jede Kollege hat (elektronischen) Zugriff auf Akten, damit er die umgeleiteten Telefonate zufriedenstellend beantworten kann.

Abbildung 2-16 Kleines Beispiel für Anforderungs-Definition [Dzid/Frei'00]

Zusammenfassend kann man sagen: Je sorgfältiger und detaillierter die Anforderungs-Definition entwickelt wurde, um so wahrscheinlicher ist es, das Ziel der Gebrauchstauglichkeit des zukünftigen SWP zu erlangen, seine Norm-Konformität zu erfüllen und somit seine Qualität zu steigern.

2.2.5 „Discount Usability“

Einen praktischen Ansatz zeigt Nielsen in seinem Aufsatz „The Discount Usability Engineering Approach“ [URL:Disc] auf . Seinen Ausgangspunkt sieht Nielsen in der Tatsache, dass in der praktischen SW-Entwicklung sw-ergonomische Vorgehensweisen immer noch keine wichtige Rolle spielen, weil zu hohe Ansprüche an sw-ergonomische Methoden gestellt werden. Seine „Discount Usability“-Methode soll die Antwort auf dieses Problem sein: Der Aufwand soll beim Usability-Engineering reduziert werden, wohlwissend, dass nicht alle Gebrauchstauglichkeitsprobleme ermittelt und behandelt werden können. *Deshalb eignet sich diese Methode nicht für grosse Projekte, sondern ist praktisch für die Gebrauchstauglichkeit-Sicherung bei kleinen Projekten.*

„Discount Usability“ basiert auf den folgenden drei Techniken:

1. Scenarios
2. Simplified thinking aloud
3. Heuristic Evaluation

Die entscheidende Rolle bei dem Vorgehen spielt das frühzeitige Einbeziehen der Endbenutzer (z. B. Erfahrungsaustausch schon am Arbeitsplatz).

Scenarien

Unter Szenarien versteht Nielsen nicht unterschiedliche Arbeitsvorgänge sondern eine bestimmte Art von Prototypen, die durch den T-Modell-Prototyping realisiert werden. Diese Prototyping-Methode wurde schon ausführlich in diesem Kapitel im Absatz „T-Modell-Prototyping“ vorgestellt.

Simplified thinking aloud

Nielsen vertritt die Ansicht, dass die Benutzer-Tests nicht mit vollem Aufwand (z. B. mit Video) durchgeführt werden sollen, sondern der Benutzer soll anhand bestimmter Aufgaben den Prototypen testen und während dessen „geradeaus“ erzählen, was er darüber denkt. Dabei soll der Experimentator die Benutzer nicht nur beobachten und einen Testbericht darüber erstellen, sondern ihm auch behilflich sein. Der Benutzer soll Idealerweise den Anwendungsbereich kennen, und der Experimentator ist am besten ein Usability-Experte.

Heuristic Evaluation

Im folgenden werden die Nielsen´sche Heuristiken und dann die Evaluierung an sich vorgestellt.

Er bietet 10 Heuristiken als Bewertungs-Kriterien für Usability (detaillierter in [URL:Disc]):

- Visibility of System Status
- Match between System and the Real World
- User Control and Freedom
- Consistency and Standards

- Error Prevention
- Recognition rather than recall
- Flexibility and Efficiency of use
- Aesthetic and Minimalist Design
- Help Users recognize, Diagnose and Recover from Errors
- Help and Dokumentation

Für diese Heuristiken sollen sog. „Evaluatoren“ Usability-Wissen besitzen, um den Prototypen zu bewerten. Die Evaluatoren werden von einem „Observer“ (Kontaktperson für Evaluatoren, Koordinator der Heuristischen Evaluation, sollte Wissen über die Anwendung oder das System haben) betreut, der sie in ihre Aufgabe einweist und ihnen alle Fragen beantwortet. Dabei sollen Idealerweise die Evaluatoren Usability-Experten sein, der „Observer“ dagegen soll im Anwendungsbereich und im Umgang mit dem Prototypen vertraut sein. Der Evaluierungsbericht soll vom Observer erstellt werden. Jeder Evaluator bekommt diesen Bericht wieder vorgelegt mit der Aufgabe, die dort aufgeführten Usability-Probleme nach den o. g. Heuristiken (Kriterien) zu bewerten. Dabei hängt es von der Aufgabe ab, ob alle Heuristiken in Betracht kommen.

Die empirischen Untersuchungen haben gezeigt, dass zwischen 3 bis 5 Evaluatoren ausreichen, um einen optimalen Kosten/Nutzen-Effekt, bei der Aufdeckung von Mängeln zu erzielen.

Die Durchführung der „Discount Usability“-Methode beinhaltet folgende Schritte und Ergebnisse:

Aktivitäten:

- Erstellen des Usability-Plans
  - ⇒ Kriterien: Ausführlichkeit der Prüfung (Analytiker- Anzahl, Benutzer, ergonomische Kriterien); Art des Prototypen (HTML-Entwürfe, "Papier"-Prototyp, ...);
  - ⇒ Anzahl der Iterationsschritte;
  - ⇒ Zuweisung der Rollen (Personalfestlegung);
  - ⇒ Zeitplan.
- Erstellen der Szenarien
  - ⇒ Feststellen der Aufgaben und der zugehörigen Nutzergruppen;
  - ⇒ Festlegen der "wichtigsten" Aufgaben, die dann für Nutzer-Test zugrunde gelegt werden.
- Erstellen des Prototypen (Prototyp + Dokumentation)
- Evaluierung des Prototypen
  - ⇒ Bericht "Heuristische Evaluierung":
    - Vorgehensbeschreibung;
    - Liste der Mängel mit Kurzbeschreibung des Mangel, Referenz zum ergonomischen Kriterium, Empfehlung zur Abhilfe, "Severity Rating".
  - ⇒ Bericht "Benutzertests":
    - Vorgehensbeschreibung;
    - Liste der Mängel mit Kurzbeschreibung des Mangels, Referenz zu ergonomischem Kriterium und Empfehlung zur Abhilfe, "Severity Rating".
  - ⇒ Usability-Bericht:
    - Vorgehensbeschreibung;
    - Anzahl der Iterationen (Heur. Eval, Benutzer-Tests, Änderung bei Prototypen);
    - Liste aller festgestellten und behobenen Mängel mit Kurzbeschreibung des Mangel, Referenz zum ergonomischen Kriterium, Empfehlung zur Abhilfe, "Severity Rating", Iterationsstufe, auf der er festgestellt wurde, Iterationsstufe, auf der er behoben wurde, Art der Abhilfe;
    - Empfehlungen bei weiterer Entwicklung des Systems (Mängel, die nicht behoben wurden)

Produkte:

- Bericht "Heuristische Evaluierung" (ggf. mehrere Versionen)

- Bericht "Benutzertests" (ggf. mehrere Versionen)
- Gesamt Usability-Bericht
- Prototyp
- Anwenderforderungen.

Der Vergleich Nielsen-Heuristiken mit Usability-Design-Prinzipien

Wie schon zu Anfang (im vorigen Absatz) angesprochen wurde, können bzgl. diesem heuristischen Vorgehen nicht alle Gebrauchstauglichkeitsprobleme ermittelt werden, was sich als ein Defizit dieser Methode erweist. Dazu kommt noch die Abneigung gegen die Einhaltung der Normen und Regeln, die Nielsen als hinderlich und abschreckend für den SW-Entwickler empfindet: *„Current user interface standards and collections of usability guidelines typically have on the order of one thousand rules to follow and are therefore seen as intimidating by developers.“*

In der folgender Tabelle (siehe Abbildung 2-17) wurde ein Abgleich der Usability-Heuristik mit den Dialoggrundsätzen der Norm ISO 9241-10 durchgeführt. Es soll deutlich werden, an welchen entscheidenden Usability-Kriterien es in der Nielsen´schen Heuristik-Methode mangelt, nämlich:

- Suitability for the task (Aufgabenangemessenheit)
- Suitability for learning (Lernförderlichkeit)
- Suitability for individualization (Selbstbeschreibungsfähigkeit)

Heuristics	Dialog Principles
- Visibility of Systems Status	⇒ Self-Descriptiveness
- Match between System and the Real World	⇒ Conformity with User Expectations
- User Control and Freedom	⇒ Controllability
- Consistency and Standards	⇒ Conformity with User Expectations
- Error Prevention	⇒ Error Tolerance
- Recognition rather than recall	⇒ Self- Descriptiveness
- Flexibility and Efficiency of use	⇒ Controllability
- Aesthetic and Minimalist Design	⇒ Conciseness
- Help Users recognize, Diagnose and Recover from Errors	⇒ Error Tolerance
- Help and Dokumentation	⇒ Self-Descriptiveness
?	• Suitability for the task
?	• Suitability for learning
?	• Suitability for individualization

Abbildung 2-17 Der Unterschied zw. Nielsen´scher Heuristik und dem Dialog-Prinzip [Dzid´00]

2.3 Die Norm ISO 13407

Die internationale Norm ISO 13407 [ISO-13407], "human centered design processes for interactive systems" (**Benutzerorientierte Gestaltung interaktiver Systeme**), beschreibt Mindestanforderungen, die an einen benutzer- und aufgabenorientierten Software-Entwicklungs-Prozess gestellt werden. Dabei werden keine detaillierten Vorschriften an die Methoden oder technische Ausstattung vorgegeben.

Dieser Standard betrifft die SW- und HW-Komponenten eines interaktiven Systems, und zwar die Bereiche Projekt-Planung und -Management. Also, die Hauptnutzer dieses Standards sollen Projekt-Manager sein, die verantwortlich für die Erfüllung ergonomischer Kriterien bei der SW-Entwicklung sind.

Der „human-centred design process“ wird als vorteilhaft angesehen, weil es den Endbenutzer in den SW-Entwicklungs-Prozess einbezieht, und zwar von Anfang an (bei der Planung), weil die Ausgewogenheit zwischen Benutzer und Technologie angestrebt wird, Wiederholbarkeit der Entwurfs-Lösungen und multidisziplinärer Entwurf möglich werden.

Die Norm [ISO-13407] für Prozesse wird als eine Ergänzung zu [ISO-9241] für Produkte angesehen (Siehe Kapitel 1, Abb.1-4 Ganzheitliches QM mittels UE).

Die Norm besteht aus 8 Punkten (Abschnitten), sowie Anhängen A, B und C. Im Punkt 2 werden die eingeführten Definitionen und der Bezug zu anderen ISO-Normen vorgestellt. Der Punkt 3 erörtert die Struktur der Norm:

- Im 4. Punkt wird der „human-centred design process“ *motiviert*.
- Im 5. Punkt werden die *Prinzipien* für einen „human-centred design“ vorgestellt.
- Im 6. Punkt wird die *Planungsphase* eines „human-centred design processes“ beschrieben.
- Der 7. Punkt ist der eigentliche *Kern* des Standards - hier werden die vier wesentlichen human-centred Aktivitäten besprochen.
- Im 8. Punkt wird *Konformitätsmöglichkeit* eines design processes zu den Standardempfehlungen angesprochen.
- Anhang: A) listet andere relevante Standards auf; B) zeigt ein *Beispiel* für eine Strukturaufstellung eines Reports über Usability-Bewertung; C) stellt ein Beispielverfahren vor zur Demonstration der Konformität mit diesem Standard.

Bestandteile des "human-centred design processes"

Der im Punkt 7 von [ISO-13407] beschriebene „human-centred design process“ besteht aus vier Aktivitäten, ihre Beispielergebnisse (*Produkte*) sind hier *kursiv* geschrieben:

- 7.2 Das Verstehen und Bestimmen des Nutzungs-Kontextes
- 7.3 Das Präzisieren von Nutzungsanforderungen
- 7.4 Die Erzeugung von Entwurfs-Lösungen
- 7.5 Das Auswerten des Entwurfs gegen die Anforderungen

Dazu siehe Abbildung 2-18.

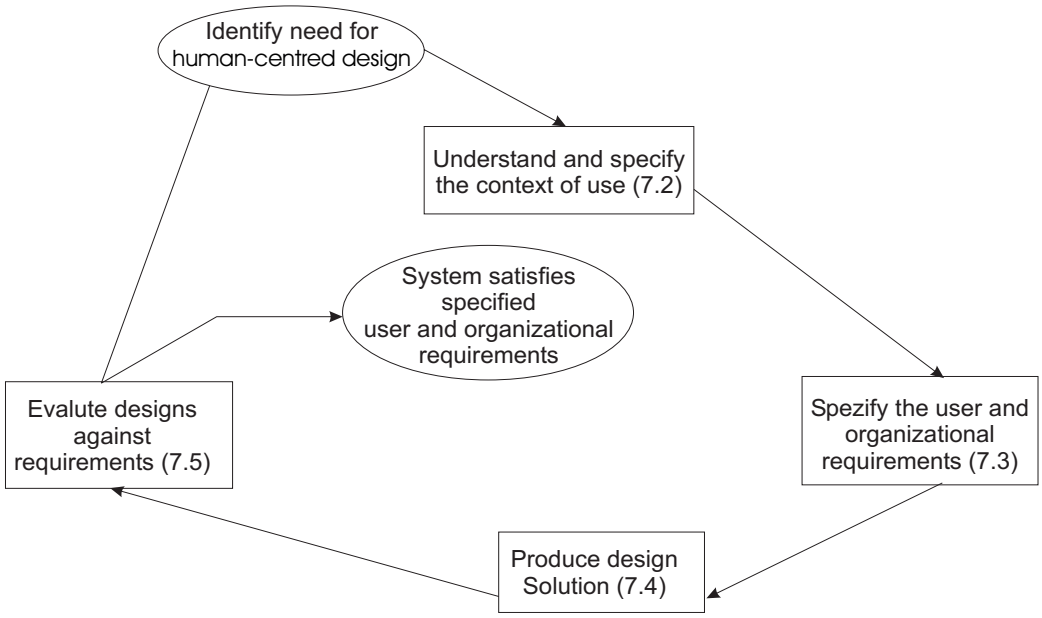

Abbildung 2-18 Abhängigkeiten innerhalb der human-centred design Aktivitäten [ISO-13407]

Die **Planung** eines System-Entwicklungsprojektes soll eine Beschreibung aller Aktivitäten und ihrer Teilaktivitäten liefern, um dann als eine Anleitung bei der Auswahl von Methoden, Techniken für die Durchführung der Aktivitäten, für Fortschrittsberichte und Befunde genutzt zu werden.

Der *human-centred design Plan* soll im Wesentlichen folgendes enthalten: Eine Liste von human-centred Aktivitäten, Verfahren zur Integration dieser Aktivitäten im Projekt, Methoden zur Ermittlung von Erfordernissen und Benutzerqualifikationen, Verfahren zur Einrichtung von Feedback/Kommunikation und Dokumentation über die Auswirkungen der Aktivitäten. Daraus resultierende Ergebnisse sollen in den gesamte System-Entwicklungsprozess einfliessen. Bei Anforderungsänderungen soll der Plan überarbeitet werden, um den Status der Aktivitäten zu reflektieren. Dieser Plan soll als Teil des gesamten System-Entwicklungsplanes angesehen werden.

Es wird empfohlen, dass der „human-centred design process“ im Anfangsstadium des Projekts (z. B. wenn ein initiales Konzept für ein Produkt formuliert wurde) gestartet werden soll, und so lange wiederholt wird, bis die Anforderungen erfüllt sind.

Als erste Aktivität des „human-centred design processes“ wird **das Verstehen und Bestimmen des Nutzungs-Kontextes (7.2)** genannt. Der Kontext, in welchem das System genutzt werden soll, wird durch die Eigenschaften des Benutzers, seine Qualifikationen, Aufgaben und seine Arbeitsausstattung definiert. Zunächst soll untersucht werden, ob ein ähnliches System schon existiert und es nur der Anpassung des System an spezielle Anforderungen oder Verbesserung des System bedarf, oder ob es Benutzer-Feedbacks gibt, die man zur Modifizierung oder Teiländerungen des System heranziehen könnte. Das Ergebnis dieser Aktivität ist eine Beschreibung der Anforderungen.

Die nächste Aktivität ist das **Präzisieren von Nutzungsanforderungen (7.3)**. Hier soll eine explizite Aussage über die Benutzeranforderungen und den organisatorischen Anforderungen, in Relation zu der Nutzungs-Kontextbeschreibung, getroffen werden. Bei der Spezifikation der Benutzeranforderungen und Anforderungen an die Organisation soll man für unterschiedliche Anforderungen Prioritäten setzen (*Prioritätenreihenfolge)*, die Menge von relevanten Benutzern festlegen *(ß-Nutzer, Endnutzer)*, Prüfkriterien für diesen Entwurf generieren (*Prüfkriterien*), klare Ziele setzen (*Ziele*), eine Liste von gesetzlichen Anforderungen bereitstellen (*relevante ISO-Normen*) und Dokumentation führen (*Dokument, Bericht*). An dieser Stelle sei festgehalten, dass in dieser Norm die Herkunft der Anforderungen nicht geklärt wurde und somit ihr Entstehungsprozess nicht betrachtet wird. Es reicht nicht aus, nur aus dem Nutzungs-Kontext die Anforderungen zu sammeln ohne zwischen Nutzungskonzept und Systemkonzept zu vermitteln.

Die **Erzeugung von Entwurfs-Lösungen (7.4)** ist die dritte Aktivität des „human-centred design processes". Sie beinhaltet fünf Teilaktivitäten:

- Zunächst soll hier ein *multidisziplinärer Input* (z. B. aus ergonomischen, psychologischen, Marketing, Design Bereichen) geschaffen werden und einfliessen, um potentielle Entwurfs-Lösungen in Betracht ziehen zu können.
- Dann sollen die Entwurfs-Lösungen konkretisiert werden und mit Hilfe von Simulationen, Modellen oder ähnlicher Methoden ausprobiert werden. Dieser *Prototyp* soll nicht nur einmal entwickelt worden sein, sondern ständig verbessert werden.
- Die *Entwurfs-Lösungen* sollen dem Benutzer vorgestellt werden, und er soll mit ihrer Hilfe bestimmte Aufgaben durchführen. (auch per Simulation)
- Von grosser Bedeutung hier ist das *Benutzerfeedback*, das wesentlich zur Optimierung des Entwurfs beiträgt.
- Des weiteren soll gewährleistet werden, dass ein Managing von verschiedenen *Versionen der Entwurfs-Lösungen* professionell betrieben wird, damit bestimmte Module aus den entsprechenden Versionen des Systems den richtigen Auswertungen zugeordnet werden können.

In dieser dritten Aktivität wird zwar vom technischen Prototyping des Systems gesprochen aber nicht vom UE-Prototyping-Prozess (Exploratives-Prototyping); dieses würde natürlich an dieser Stelle viel zu spät bedacht, wenn es nun überhaupt eingesetzt würde. Das UE-Prototyping und sein Einsatz wird im Kapitel 4 genauer ausgearbeitet.

Der essentielle Schritt im „human-centred design process" ist die **Evaluierung (7.5)**. Diese Aktivität soll in allen Stadien des „systems-life-cycle" durchgeführt werden können. Sie kann zur Feedback-Erhebung, mit dessen Hilfe man den Entwurf verbessern kann, zum Präzisieren der Ansteuerung von Benutzerziele, und zur Überwachung der Systemqualität ausgebeutet werden. Ihre Teilaktivitäten sind die Erstellung der Planung (*Plan*), Feedback-Erhebung (*Feedbackdokument*), Fokussierung der Ziele (*realistische Ziele*), Spezifikation von Gültigkeitsbereichen, Erstellung von Resultaten (*Resultatenbericht*).

Zusammenfassend kann man sagen, dass diese Norm [ISO-13407] einerseits sehr hilfreich ist, um sich bei der SW-Entwicklung überhaupt Gedanken über Benutzerbeteiligung am Geschehen zu machen und seine Erkenntnisse in den Entwicklungs-Prozess einfliessen zu las-

sen. Anderseits aber ist es für einen Usability-Engineer recht schwierig in der Praxis mit diesen allgemeinen Formulierungen etwas anfangen zu können, weil die meisten Aussagen in einer Empfehlungsform allgemeiner Art sind („man kann“). Eine Interpretation der allgemeinen Empfehlungen ist nicht immer möglich, weil sogar das Beispiel am Anhang kaum was konkretes enthält. Z. B ist es offensichtlich, dass die Entwicklung der Anforderungen gar nicht berücksichtigt worden ist, man spricht nur von ihrer Analyse.

Fazit: Zur Umsetzung der Norm sollte sich der Usability-Engineer ein eigenes Prozessmodell entwickeln. Die Norm gibt dem Usability-Engineer lediglich einige Hinweise, worauf bei einem Prozessmodell zu achten ist. Auf diese Weise kann Konformität mit der Norm nachgewiesen werden (dazu siehe Punkt 8 „Conformity“). Die Konformitätsausage soll sich auf den Nachweis von "human-centred design activities" beziehen, soweit sie dokumentiert wurden. Zum Beispiel:

- Nachweis der Analyse des Nutzungs-Kontextes. *Hier fehlt aber der Nachweis über die Validierung der Kontextdaten.*
- Nachweis der benutzerorientierten Ziele der Gestaltung, und der Anforderungen. *Hier fehlt aber der Nachweis über die Entwicklung dieser Anforderungen sowie deren Validierung.*

3 Das V-Model

Die traditionellen Ingenieurdisziplinen zeichnen sich insbesondere dadurch aus, dass meist Standards zur Spezifikation der Konstruktion und der Ergebnisse vorgegeben sind und bereits vorhandene Methoden und Vorgehensmodelle weiter verwendet werden können. In dieser Diplomarbeit ist der Fokus auf das Vorgehensmodell *V-Modell* (**VM**) ausgerichtet - Standard zur Beschreibung des IT-Entwicklungsprozesses, um diesem Ingenieur Gedanken treu zu bleiben. Ausserdem macht es seine universelle Einsetzbarkeit bei der SW-Entwicklung, seine Allgemeingültigkeit, [ISO-12207] Konformität lohnenswert, gerade dieses Modell um ein UE-Prozess-Modell zu erweitern.

Zunächst werden in diesem Abschnitt die für die Diplomarbeit relevanten Aspekte des V-Modells vorgestellt. Im nächsten Kapitel wird der VM-Formalismus für das UE-Teil des SW-Entwicklungsprozesses genutzt, um es zu beschreiben, damit man dann das V-Modell um den UE-Prozess erweitern kann. Die formale Darstellung und Beschreibung des V-Modells erfolgt unter anderen anhand der folgenden Quellen [Drös´00], [Vers´00], [KBSt´97], [URL:V-Model] , [URL:12207]:

Das *V-Modell* ist ein Vorgehensmodell für die SW-Entwicklung, das im Auftrag des Bundesministeriums für Verteidigung entwickelt und 1992 in einer überarbeiteten und mit dem Bundesministerium des Innen abgestimmten Version herausgegeben wurde (mittlerweile gibt es die 1997 Version). Es war als Grundlage für Ausschreibungen und Verträge gedacht, kann aber innerhalb eines Unternehmens für Prozessabläufe auch verwendet werden [URL:V-Model].

Das V-Modell beschreibt den Entwicklungsprozess anhand von Aktivitäten und Produkten (Ergebnissen), die im Rahmen dieser Aktivitäten erstellt wurden. Es ist in vier Submodelle gegliedert: Systemerstellung **SE**, Projektmanagement **PM**, Qualitätssicherung **QS,** Konfigurationsmanagement **KM**. Das V-Modell regelt die Entwicklung sowie die Wartung und Änderung von Systemen, deren Aufgabenerfüllung vorwiegend durch den Einsatz von IT realisiert wird. Dies geschieht durch die einheitliche, transparente und verbindliche Vorgabe von Aktivitäten und Produkten, die bei der IT-Systemerstellung und den begleitenden Tätigkeiten für **QS**, **KM** und technisches **PM** anfallen.

3.1 Überblick über das V-Modell

Alle Tätigkeiten und Ergebnisse der Entwicklung und der Wartung/Änderung von IT-Systemen werden durch den Entwicklungsstandard VM für IT-Systeme geregelt. Diese Standardisierung ist für drei Bereiche (Vorgehensweise, anzuwendende Methoden, funktionale Anforderungen an einzusetzende Werkzeuge) vorgegeben [URL:V-Model]:

- *Vorgehensmodell*: Beschreibung der Aktivitäten und Produkte als Antwort auf die Frage, "***was***" getan wird,
- *Methodenzuordnung*: Beschreibung der Mindestanforderungen an Methoden zur Beantwortung der Frage, "***wie***" es getan wird,
- *Funktionale Werkzeuganforderungen*: Standardisierter Kriterienkatalog zur Werkzeugauswahl, der angibt, "***womit***" etwas getan wird.

In dieser Diplomarbeit wird der Bereich des *Vorgehensmodells* vorgestellt und untersucht. Das Vorgehensmodel besteht aus den folgenden Teilen [URL:V-Model]:

- **Regelungsteil** ist die einheitliche und verbindliche Vorgabe von Aktivitäten und Produkten, die bei der IT-Systemerstellung und den begleitenden Tätigkeiten für Qualitätssicherung QS, Konfigurationsmanagement KM und technisches Projektmanagement PM anfallen.
- **Handbuchsammlung** enthält Erläuterungen zu verschiedenen Themen, wie: Zusammenhang zwischen Geschäftsprozessoptimierung und dem V-Modell, Hardwareerstellung, das V-Modell in einer ISO- und AQAP-Umgebung, Berücksichtigung objektorientierter Sprachen, Rollenkonzept im VM, Reverse-Engineering, Anwendung des V-Modells und der ITSEC, Sicherheit und Kritikalität, Szenarien, Tailoring und projektspezifisches V-Modell, Einordnung des VM in sein Umfeld.

  Die Handbuchsammlung wird bei Bedarf um weitere Handbücher ergänzt (unabhängig von der Herausgabe einer neuen Version des VM). Die Handbücher sollen Hilfestellung bei der Arbeit mit dem VM geben. Sie haben dementsprechend keinen Regelungscharakter.
- **Behördenspezifische** Ergänzungen (der Vollständigkeitshalber erwähnt, aber für die Diplomarbeit nicht relevant)

Der relevante Teil des V-Modells für die Diplomarbeit ist der **Regelungsteil**. Er regelt die Gesamtheit aller *Aktivitäten* und *Produkte* und *Produktzustände*, sowie *logische Abhängigkeiten* zwischen Aktivitäten und Produkten. Der Standard beschreibt den Entwicklungsprozess aus funktionaler Sicht. Das V-Modell geht nicht auf spezielle Organisationsformen ein, da es in unterschiedlichsten Einrichtungen/Firmen zum Einsatz kommen soll (z. B. Verkehrssteuerung, Chemie-Industrie, Banken, Lagerbetrieb u. ä.). Für die Durchführung eines Projekts müssen daher für die im VM dargestellten Aufgaben (Aktivitäten) Bearbeiter bzw. Organisationseinheiten festgelegt werden. Das VM versteht sich von der Konstruktion her also als ein Baukasten. Daher ist die Anpassung des VM auf ein konkretes Umfeld notwendig (**Tailoring**). Prinzipiell handelt es sich um eine zwei-stufige Anpassung: *Anpassung an das Unternehmen* (Terminologie, Projekttypen, Aktivitätenbeschreibung, echtzeit- oder datenorientierte Anwendung, Rollen, Zuständigkeiten usw.) und *Anpassung an das Projekt* (Auswahl der Aktivitäten und Produkte).

Innerhalb des VM werden gewisse Aktivitäten zusammengefasst, weil sie jeweils aus einer bestimmten Sicht ein für sich abgeschlossenes Modell repräsentieren. Ein derartiges Modell wird als **Submodell** bezeichnet. Das VM also ist in vier Submodelle gegliedert [URL:V-Model]:

- Submodell Systemerstellung **SE** : Hier wird die Erstellung eines IT-Systems geregelt. Die Erstellung der Software erfolgt unter Berücksichtigung der Schnittstelle zur Hardware.
- Submodell Qualitätssicherung **QS** : Hier werden die Massnahmen zur Erreichung der Qualitätsziele und zur Vermeidung von Qualitätsrisiken sowie die analytischen Massnahmen zum Nachweis der geforderten Qualität beschrieben. Das Submodell QS umfasst: Planung von präventiven, konstruktiven und analytischen Massnahmen; Vorbereitung, Sicherstellung, Durchführung und Berichtswesen für die analytischen Massnahmen, und bezieht sich auf Organisation, Prozesse und Produkte.

- Submodell Konfigurationsmanagement **KM** : Hier wird die Verwaltung der während der Systementwicklung anfallenden Produkte aller vier Submodelle geregelt. Ihre verschiedenen Versionen und Beziehungen zueinander werden konsistent gehalten, Änderungen überwacht und Zugriffe dokumentiert.
- Submodell Projektmanagement **PM** : Hier werden alle technischen Aspekte der Projektabwicklung geregelt. Dazu gehören Planung, Steuerung und Kontrolle aller projektinternen Tätigkeiten sowie projektinterne und projektexterne Kommunikation (z. B. mit Auftraggeber, Anwender). Nicht abgedeckt sind jedoch System-Management und Finanzplanung.

Der Zusammenspiel der Submodelle ist in Abbildung 3-1 dargestellt als eine hierarchische Erzeugnisstruktur.

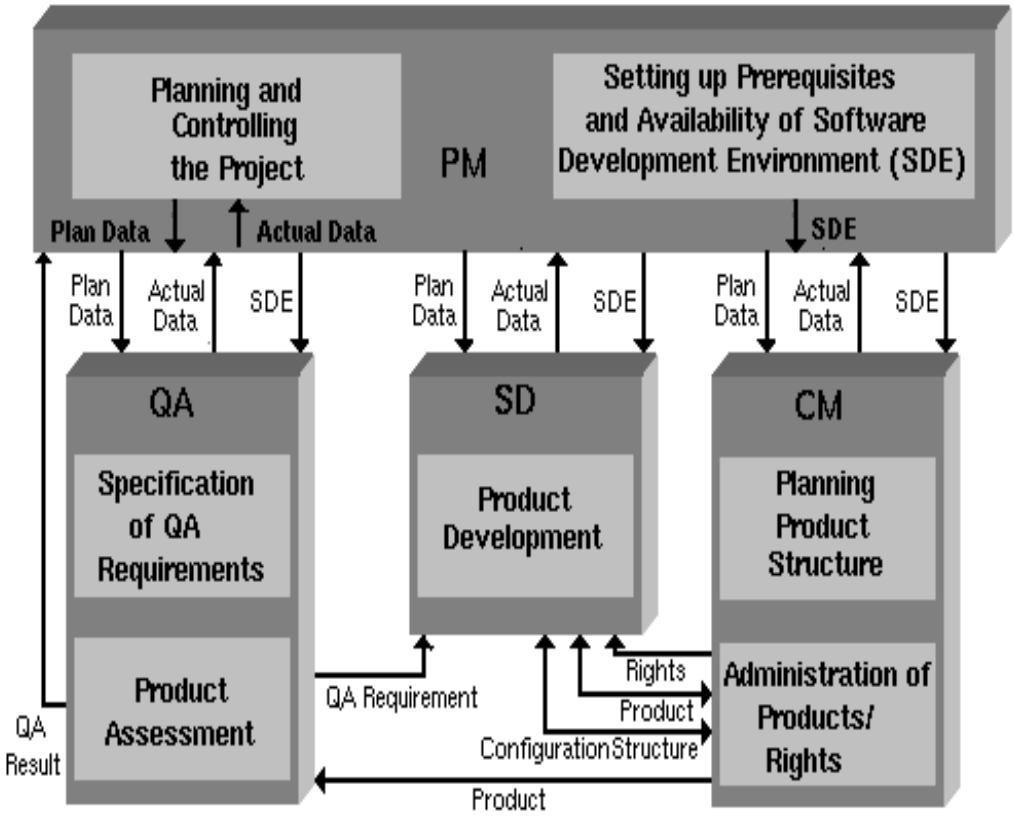

Abbildung 3-1 Zusammenspiel der Submodelle [URL:V-Model]

Die Abbildung 3-2 zeigt, wie die Produkte der vier Submodelle SE, QS, KM und PM in die hierarchische Erzeugnisstruktur einzuordnen sind.

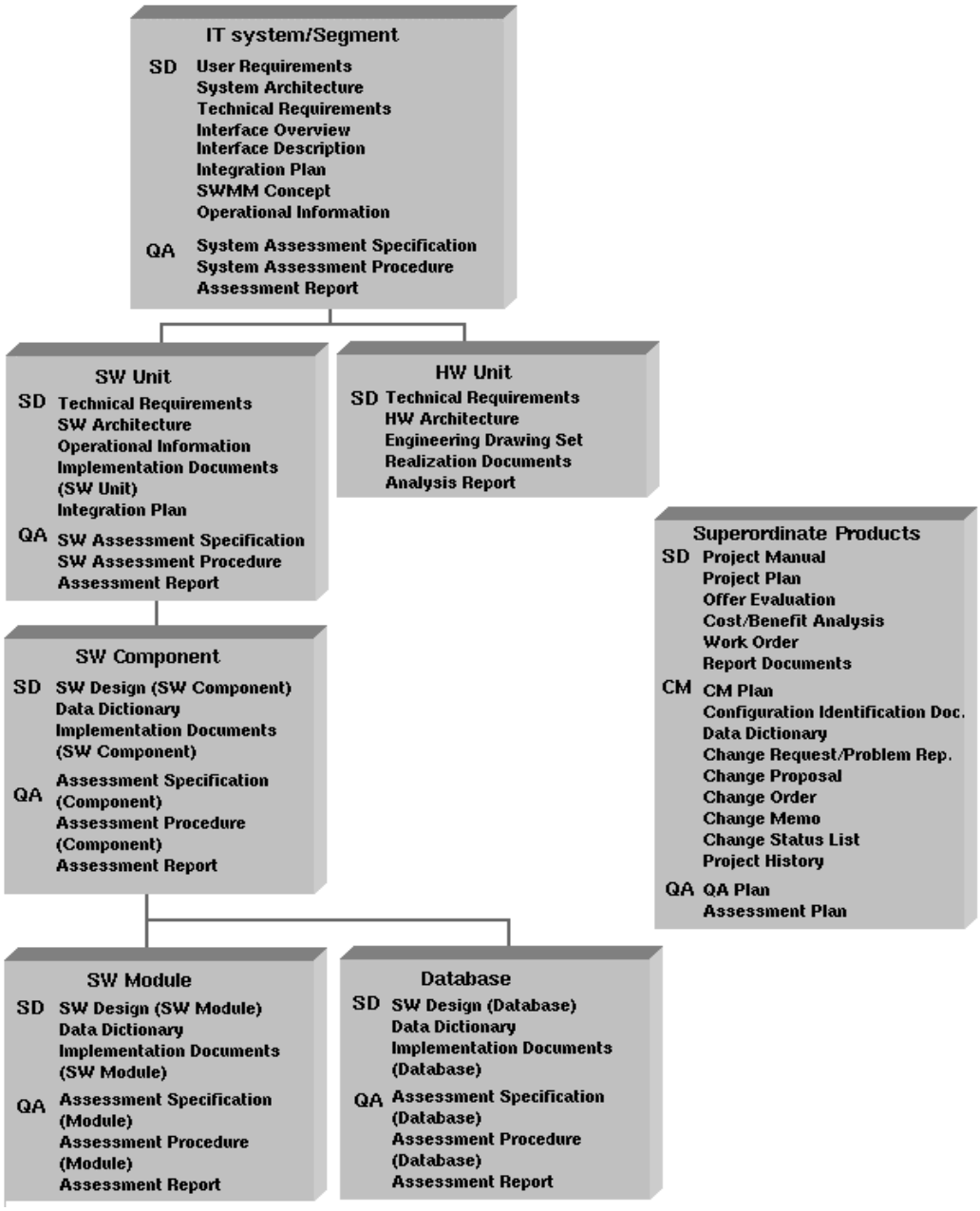

Abbildung 3-2 Produktbaum [URL:V-Model]

Der für die Diplomarbeit relevante Teil ist das Submodell **Systemerstellung** (**SE)**, in dem die Erweiterung um das Usability-Engineering UE stattfinden wird (dazu siehe Kapitel 4).

3.1.1 Definitionen, Strukturen und das organisatorische Umfeld

In diesem Unterabschnitt werden Definitionen, Beschreibungsmittel und Grundkonzepte vorgestellt, die im V-Modell verwendet werden.

Die Grundelemente des V-Modells sind die Aktivitäten und Produkte, die während des IT-Systementwicklungsprozesses durchgeführt bzw. bearbeitet werden [URL:V-Model].

Aktivitäten und Produkte

Unter *Aktivität* versteht man im VM eine Tätigkeit im Rahmen des IT-Systementwicklungsprozesses, die hinsichtlich ihres Ergebnisses und ihrer Durchführung genau beschrieben werden kann. Aktivitäten können aus einer Reihe festgelegter *Teilaktivitäten* bestehen, wenn jede dieser Teilaktivitäten ihrerseits definierte *Zwischenergebnisse* aufweist. Die Aktivitäten der obersten Detaillierungsebene werden *Hauptaktivitäten* genannt.

Das *Produkt* im VM ist der Bearbeitungsgegenstand bzw. das Ergebnis einer Aktivität. Analog der Zerlegung von Aktivitäten in Teilaktivitäten kann sich die Zerlegung von Produkten in "Teilprodukte" (z. B. einzelne Kapitel eines Dokuments) ergeben.

Dabei kann eine Aktivität die Erstellung eines Produkts, eine Zustandsänderung eines Produkts oder die inhaltliche Änderung eines Produkts zum Gegenstand haben.

Für jede Aktivität gibt es eine *Aktivitätenbeschreibung* als Arbeitsanleitung, der bei der Ausführung der Aktivität zu folgen ist. Die Aktivitätenbeschreibung erfolgt nach einem festen Muster, dem *Aktivitätenschema*. Wenn eine Aktivität in Teilaktivitäten zerlegt wird und logische Abhängigkeiten zwischen Teilaktivitäten und Produkten herausgestellt werden müssen, enthält das Aktivitätenschema zusätzlich eine grafische Darstellung der Teilaktivitäten.

Für jedes Produkt gibt es eine *Produktbeschreibung*, welche die Inhalte des Produkts definiert. Die Produktbeschreibung erfolgt nach einem festen Muster, dem **Produktmuster**: Für jedes Produkt wird eine kurze Zusammenfassung des Produktinhaltes eine Auflistung der inhaltlichen Gliederungspunkte des Produktes angegeben. Anforderungen an das Layout werden im Rahmen des VM nicht gestellt. Die Feinstrukturierung der betroffenen Dokumente soll von Fall zu Fall vereinbart werden. Dabei ist die Abdeckung der erforderlichen Inhalte nachzuweisen.

Produktzustände

Im V-Modell ist zu berücksichtigen, dass gewisse Produkte im Entwicklungsprozess verschiedene Zustände durchlaufen. Der Wechsel von einem Zustand in einen anderen wird dabei grundsätzlich durch eine Aktivität ausgelöst. Alle Zustandsübergänge werden bei den entsprechenden Aktivitäten selbst angegeben (siehe Aktivitätenschema).

Hier folgende Zustandsarten [URL:V-Model]:

- "geplant", das Produkt ist in der Planung vorgesehen. Dies ist der Eingangszustand für alle Produkte.
- "in Bearb.", das Produkt wird bearbeitet. Es befindet sich entweder im (privaten) Entwicklungsbereich des Entwicklers oder unter Kontrolle des Entwicklers innerhalb der Produktbibliothek.
- "vorgelegt", das Produkt ist aus der Sicht des Erstellers fertig und wird unter Konfigurationsverwaltung genommen. Es wird einer QS-Prüfung unterzogen. Wird das Produkt in QS abgelehnt, so geht es wieder in den Zustand "in Bearbeitung" zurück, andernfalls

rückt es in den Zustand "akzeptiert" vor. Vom Zustand "vorgelegt" ab, kann der Ersteller nur unter Fortschreibung der Versionsangabe Modifikationen durchführen.

- "akzeptiert", das Produkt wurde in QS überprüft und freigegeben, und darf nur mehr innerhalb einer neuen Version geändert werden.

Die mindestens geforderten Zustandsübergänge sind in Abbildung 3-3 dargestellt.

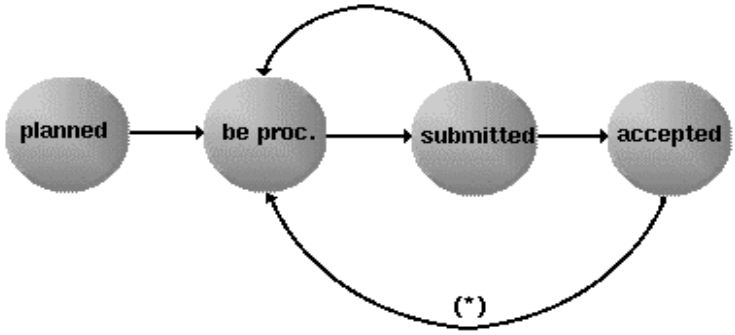

Abbildung 3-3 Zulässige Zustandsübergänge von Produkten [URL:V-Model]

Aktivitätenschema

Nach folgendem Muster werden Aktivitäten beschrieben: *Aktivitäts-Name*, *Produktfluss*, *Abwicklung* [URL:V-Model] :

Produktfluss: Alle Produkte oder Teilprodukte, die in die (Teil-) Aktivität eingehen oder von ihr modifiziert oder neu erstellt werden, werden aufgelistet und ihre Behandlung beschrieben. Das geschieht in Form einer Tabelle (siehe tab. Musterbeispiel für einen Prodfluss), die entsprechend ausgefüllt wird. Teilprodukte werden in Punktnotation dargestellt. So bedeutet z. B. "Systemarchitektur.Anforderungszuordnung" dass Anforderungszuordnung ein Teilprodukt des Produkts "Systemarchitektur" ist.

PRODUKTFLUSS **SE2.2** :

von Aktivität	**von Zustand**	*Produkt*	**nach Aktivität**	**nach Zustand**
SE1	akzeptiert	*Anwenderforderungen*	-	-
SE2.1	in Bearb.	*Systemarchitektur*	-	-
-	-	*Systemarchitektur.IT-Sicherheitskonzept*	SE2.1	in Bearb.

Abbildung 3-4 Musterbeispiel für einen Produktfluss [Drös´00]

Für jedes (Teil-)Produkt (Spalte 3), das durch die zu beschreibende Aktivität betroffen ist, wird der Zustand zu Beginn der Aktivität (Spalte 2) und der Zustand nach Beendigung der Aktivität (Spalte 5) angegeben. Hat die Aktivität keinen Einfluss auf den Produktzustand oder existiert kein Produktzustand, so ist dies in der Tabelle durch einen Strich angegeben. *Inputprodukte* einer Aktivität werden kenntlich gemacht, indem die Spalten "von Aktivität" und "von Zustand" ausgefüllt sind, die "nach Aktivität" und der "nach Zustand" sind mit Strich versehen. Bei *Outputprodukten* sind nur die "nach Aktivität" und der "nach Zustand" vermerkt. Hat ein Produkt sowohl "von" als auch "nach" Einträge, so bedeutet das, dass es in der betreffenden Aktivität modifiziert wird. Alle in der Aktivität neu erstellten *Outputprodukte*, deren Outputzustand "in Bearb." ist, müssten modellgemäss den *Inputzustand* "geplant" haben. Um jedoch *Input- und Outputgangsprodukte* optisch besser unterscheiden zu können, ist der Inputzustand durch "-" ersetzt und als "geplant" zu interpretieren.

Bei jedem (Teil-) Produkt wird auch vermerkt, *von* welcher Aktivität das Produkt kommt (Spalte 1) und an welche Aktivität *nach* das Produkt weitergegeben wird (Spalte 4). Gibt es zu einem (Teil-) Produkt keine "von" oder "nach" Aktivität, so ist dies in der Tabelle durch einen Strich angegeben.

Falls in unterschiedlichen (Teil-) Aktivitäten (z.B. Aktivität QS 2.2 Prüfumgebung definieren) Teilprodukte eines Produkts entstehen, ergibt sich die Notwendigkeit, das Produkt durch Integration der Teilprodukte zusammenzustellen. Dies erfolgt in der Aktivität, in der das zuletzt entstehende Teilprodukt eines Produkts erstellt wird. Im *Produktfluss* wird dies dadurch zum Ausdruck gebracht, dass in der Spalte "nach-Aktivität" auf nachfolgende Hauptaktivitäten verwiesen wird. Bei Produkten, die nicht weiter fortgeschrieben werden, wird in der Spalte "nach-Zustand" der Zustand "vorgelegt" zugeordnet.

Es ist hier anzumerken, dass Massnahmen zur Qualitätssicherung und zum Konfigurationsmanagement indirekt durch die Zustandsänderungen angedeutet werden.

Abwicklung: Es erfolgt eine verbindlich Festlegung, wie bei der Durchführung der Aktivität vorzugehen ist. Bei einer Hauptaktivität sind zu Beginn des Abwicklungstextes ihre Teilaktivitäten, deren Abhängigkeiten untereinander sowie deren Ergebnisse grafisch dargestellt. Dabei gilt folgende Festlegung: Ist zu Teilaktivitäten kein Ergebnisprodukt angegeben, so bedeutet dies, dass in mehreren in der Abbildung aufeinanderfolgenden Teilaktivitäten an Teilen eines gemeinsamen Produktes gearbeitet wird.

Sonst: In Ergänzung zu den Abwicklungen können Unterabschnitte *Empfehlung* und *Erläuterung* gegeben werden. Im Gegensatz zur oben angeführten verbindlichen Festlegung

sind die ausgesprochenen Empfehlungen nicht bindend. Die Erläuterungen sollen die Aufgabenstellung und die Abwicklung der Aktivität verdeutlichen.

Das System-Begriff

Im VM wird unter einem System folgendes verstanden: „In seiner Anwendung bezieht sich das V-Modell prinzipiell auf den Systembegriff, wie er in der [ISO-12207] festgelegt ist: Ein *System* ist dort definiert als ein einheitliches Ganzes, das aus einem oder mehreren Prozessen, Hardware, Software, Einrichtungen und Personen besteht, das die Fähigkeit besitzt vorgegebene Forderungen oder Ziele zu befriedigen. Für das V-Modell werden nur die Systeme betrachtet, deren Aufgabenerfüllung vorwiegend durch den Einsatz von IT realisiert wird. Dazu zählen neben den IT-Systemen auch IT-Anteile in anderen Systemen. Im V-Modell wird dafür abkürzend der Begriff "IT-System" verwendet. Ein IT-System in diesem Sinne kann Teil eines übergeordneten Systems sein“ [URL:V-Model].

Die Verkürzung des Systembegriffs auf „IT-System“ ist im UE problematisch, weil UE in der Tradition der Gestaltung von Mensch-Maschine-System steht. Hier nehmen Benutzer einen nicht zu übersehenden Platz im System ein.

Das V-Modell beschreibt die Entwicklung von IT-Systemen inkrementell auf verschiedenen Ebenen, angefangen von *System-Ebene* bis hin zu *Modul-Ebene*. Das *System* ist eine Funktionseinheit der obersten Ebene aus der Sicht des V-Modells.

„Die *Erzeugnisstruktur* des V-Modells definiert, aus welchen generischen Bausteinen ein zu entwickelndes System grundsätzlich besteht.

Die *Systemarchitektur* beschreibt den statischen Aufbau eines Systems als vernetzte Struktur mit den Elementen der generischen Erzeugnisstruktur. Dynamische Aspekte des Systems werden über die Beschreibung der Funktionsweise und dem Zusammenwirken der Elemente der Erzeugnisstruktur über deren Schnittstellen dargestellt„[URL:V-Model].

Die Abbildung 3-5 zeigt die hierarchisch gegliederte Erzeugnisstruktur des Systems. Ausserdem ist gekennzeichnet, welche Entwicklungstätigkeiten durch das V-Modell geregelt sind.

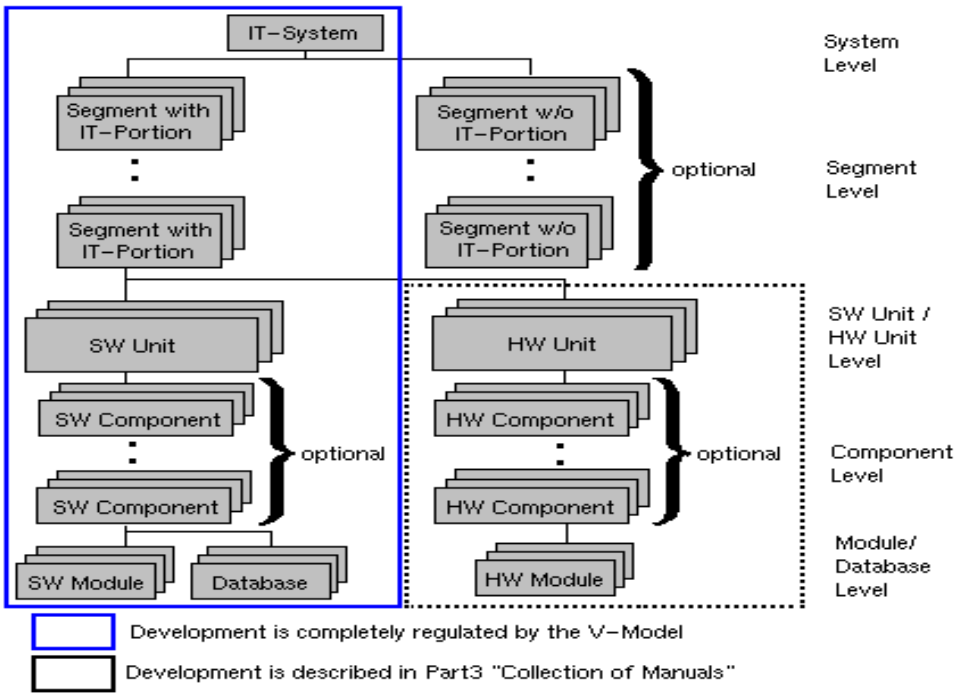

Abbildung 3-5. Erzeugnisstruktur und Anwendungsbereich des V-Modells [URL:V-Model]

Im folgenden wird beschrieben, aus welchen Bestandteilen sich ein *System* im Sinne des V-Modells zusammensetzt: „Ein System gliedert sich in **Segmente**, die unterschieden werden in solche mit IT-Anteilen (Segmente mit IT-Anteil) und solche ohne IT-Anteile (Segmente ohne IT-Anteil). Segmente können ihrerseits wieder aus mehreren Segmenten zusammengesetzt sein.

Segmente werden weiter in *SW-Einheiten* und *HW-Einheiten* gegliedert. Systeme können sich auch direkt in SW-Einheiten und HW-Einheiten gliedern. Es ist auch der Fall möglich, dass sich ein System sowohl in Segmente als auch direkt in SW-Einheiten/HW-Einheiten gliedert [URL:V-Model]:

SW-Einheiten (z. B. Lademodule) und **HW-Einheiten** (z. B. Baugruppen oder kleinere Geräte) stellen aus der Sicht der Systemarchitektur elementare Objekte dar. Die Entwicklung von SW-Einheiten wird durch den *Regelungsteil* des V-Modells abgedeckt; die Entwicklung der HW-Einheiten wird in Hardwareerstellung von *Handbuchsammlung* behandelt (der HW-Teil ist für die Diplomarbeit nicht relevant, nur Vollständigkeitshalber erwähnt).

SW-Einheiten bestehen aus **SW-Komponenten**, wobei SW-Komponenten entweder selbst wieder aus SW-Komponenten niedrigerer Ordnung, SW-Modulen und/oder Datenbanken zusammengesetzt sein können.

SW-Komponenten sind abgeschlossene Softwareeinheiten einer höheren Ebene.

SW-Module sind die kleinsten zu programmierenden Softwarebausteine einer SW-Einheit.

Datenbanken dienen der Beschreibung, Speicherung und Wiedergewinnung von Daten.

HW-Einheiten bestehen aus HW-Komponenten, wobei HW-Komponenten entweder selbst wieder aus HW-Komponenten und/oder HW-Modulen zusammengesetzt sein können.

HW-Module sind die kleinsten Bausteine einer HW-Einheit.

Das Tailoring

Wie zum Anfang schon angesprochen ist VM ein Baukasten, der sich durch Allgemeingültigkeit auszeichnet. D. h. die enthaltenen technischen Regelungen müssen an die konkrete Projektsituation angepasst werden. Diese Anpassung für ein Projekt (eben Tailoring genannt) kann durch Streichung oder Zusammenfassung von Aktivitäten oder Produkten durchgeführt werden - Ergänzungen müssen kenntlich gemacht werden. Im Anschluss daran müssen die organisatorischen Zuordnungen der so gefundenen Aktivitäten erfolgen. In der Handbuchsammlung (siehe 4.2) gibt es u. a. auch Anpassungsvorschläge für verschiedene Projekt-/ Vorhabenstypen im Rahmen des "Standardisierten Vortailoring" [URL:V-Model].

3.1.2 Die Aktivitäten des Submodells SE

Das Submodell SE ist ein V-förmig strukturiertes Phasenmodell (daher auch der Name V-Modell) mit einer schrittweise Verfeinerung der Systemanforderungen (top-down) und anschliessender Integration der erstellen Komponenten (bottom-up) (dazu siehe Abbildung 3-6). Wie alle Submodelle besteht es aus Aktivitäten sowie der im Rahmen der Aktivitäten erstellten Produkte, die dann jeweils noch ausführlicher beschrieben sind. Die Beschreibung einer Aktivität besteht aus dem tabellarischen Produktfluss sowie der in Prosa beschriebenen Abwicklung. Dabei beschreiben die drei Submodelle Qualitätssicherung **QS**, Projektmanagement **PM** und Konfigurationsmanagement **KM** die begleitenden Aktivitäten in einem Entwicklungsprojekt, die Entwicklung selbst aber wird im Submodell Systemerstellung **SE** durchgeführt [URL:V-Model].

Die Aufteilung des Submodells **SE** ist dominiert durch Aktivitäten auf zwei verschiedenen Ebenen, nämlich: *System-(Segment-)Ebene* und Ebene der Software- und Hardwareeinheiten (*SW-Einheiten/HW-Einheiten*). Innerhalb der beiden Ebenen wird jeweils zwischen dem Schritt der Anforderungsanalyse und dem Schritt des Entwurfs unterschieden.

Die Aktivitäten des Submodells **SE** sind [URL:V-Model]:

- SE1 : System-Anforderungsanalyse
- SE2 : System-Entwurf
- SE3 : SW-/HW-Anforderungsanalyse
- SE4-SW : SW-Grobentwurf
- SE5-SW : SW-Feinentwurf
- SE6-SW : SW-Implementierung
- SE7-SW : SW-Integration
- SE8 : System-Integration
- SE9 : Überleitung in die Nutzung

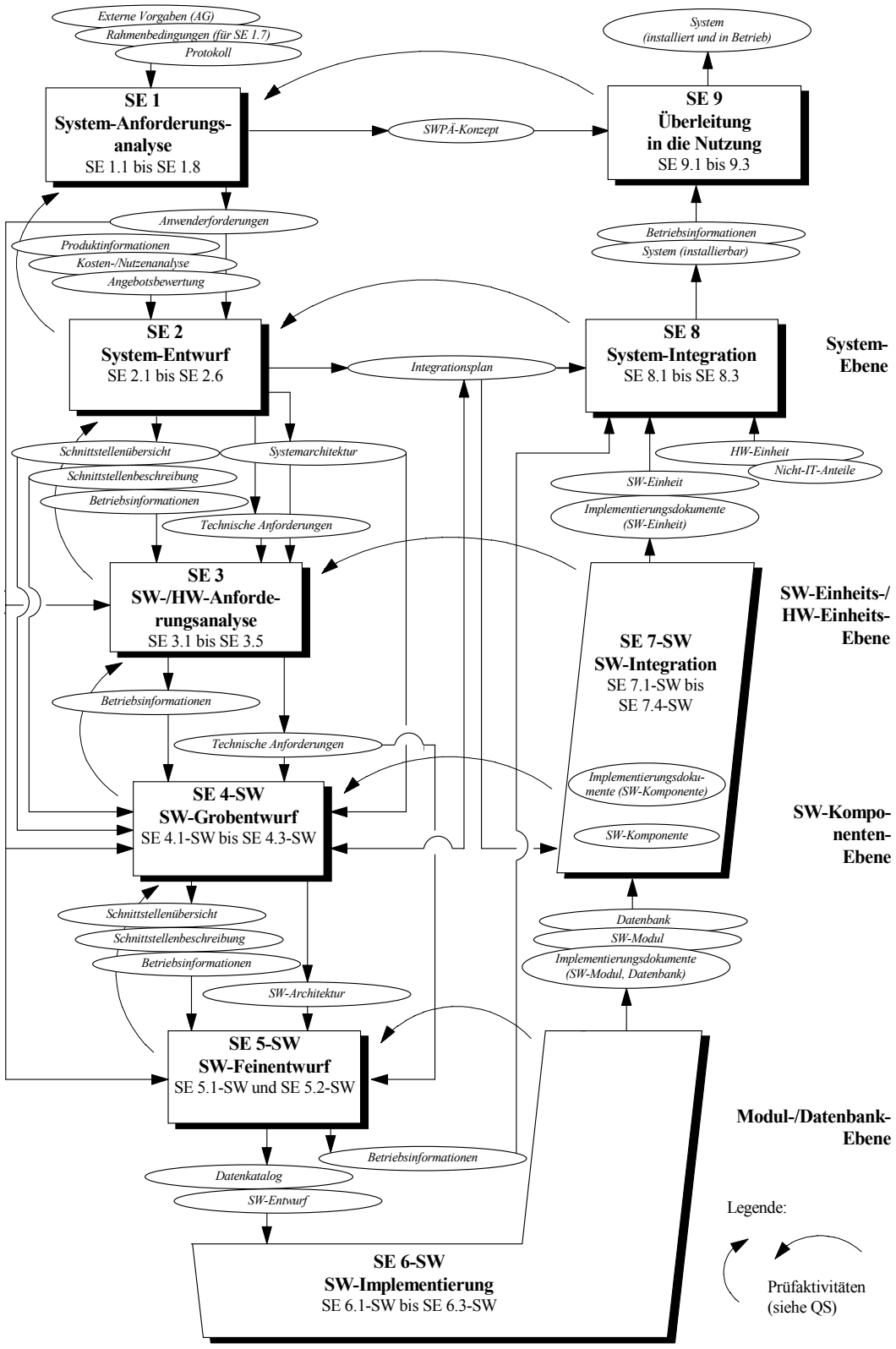

Abbildung 3-6 Funktionsüberblick Submodell SE [Drös´00]

Die Formulierungen in der Abwicklung des Regelungsteils gehen davon aus, dass standardmässig inkrementell entwickelt wird; d.h. die Regelungen des Submodells SE-Systemerstellung sind so zu verstehen, dass ein System in Stufen entwickelt und evtl. ausgeliefert und betrieben wird.

3.2 Software-Lifecycle-Processes in ISO 12207

Der Internationale Standard [ISO-12207] bezieht sich auf *Software-Lifecycle-Processes* und ist ein weltweit akzeptierter Standard. Er ist zwar ungeeignet für den direkten Einsatz in einem konkreten Projekt, gibt jedoch einen Rahmen vor, der es erlaubt, nationale Standards einzuordnen oder entsprechende Detaillierungen der Prozesse vorzunehmen, um in der Praxis anwendbare Standards zu erzielen. Weiter beinhaltet die [ISO-12207] Definitionen, die als Basis für eine gemeinsame Terminologie auch für nationale Standards dienen können [URL:12207].

Da in der Diplomarbeit das VM um das besagte UE-Prozess-Modell (Kapitel 4) ergänzt wird, müssen beide im Hinblick auf die [ISO-12207] konform sein. Im folgenden wird zunächst ein Überblick über die [ISO-12207] gegeben, und es wird dargestellt, wie die Prozesse der [ISO-12207] durch das V-Modell realisiert werden.

3.2.1 Überblick über die ISO 12207

Im Folgenden werden der *Anwendungsbereich* des Standards ISO 12207, der *Ablauf* eines Projektes, die *Auftragsvergabe* und die *Beteiligung* verschiedener Stellen bei einer Durchführung nach [ISO-12207] erläutert [URL:12207]:

Aufgabengebiete: Die [ISO-12207] kann für die *Beschaffung*, die *Lieferung*, die *Entwicklung*, den *Betrieb* und die *Wartung* von Systemen/SW angewendet werden, unabhängig davon, ob die Entwicklung organisationsintern durchgeführt oder extern vergeben wird. Sie gilt sowohl für SW als eigenständiges Produkt, als auch für SW als Bestandteil eines Gesamtsystems. Neben der *SW-Entwicklung* betrachtet der Standard auch Aspekte wie *Qualitätssicherung* und *Projektmanagement*.

Diese Norm muss für den Einsatz in einem konkreten Projekt auf die jeweiligen Erfordernisse zugeschnitten werden (Tailoring), dabei werden nicht relevante Aufgaben „gestrichen“.

Der Standard [ISO-12207] schreibt kein Entwicklungsmodell vor, sondern liefert einen *Software Lifecycle*, in welchen im Projektfall ein ausgewähltes Entwicklungs-/ Vorgehensmodell einzubinden bzw. darauf abzubilden ist. Mit Hilfe von *Prozessen*, die weiter in *Aktivitäten* zerlegt werden, beschreibt die ISO den *SW-Lifecycle*. Die Aktivitäten werden weiter in *Aufgaben* (tasks) zerlegt, die aber vollständig noch im Detail ausgeführt werden - nachdem kein konkretes Vorgehensmodell spezifiziert werden soll. Die Norm enthält keinen *Regelungen* (wie das VM) bzgl. anzuwendender Methoden oder zu erstellender Produkte. Diese Festlegungen bleiben dem Anwender der [ISO-12207] vorbehalten.

Die Abbildung 3-7 gibt einen Überblick über die in der [ISO-12207] geregelten Prozesse.

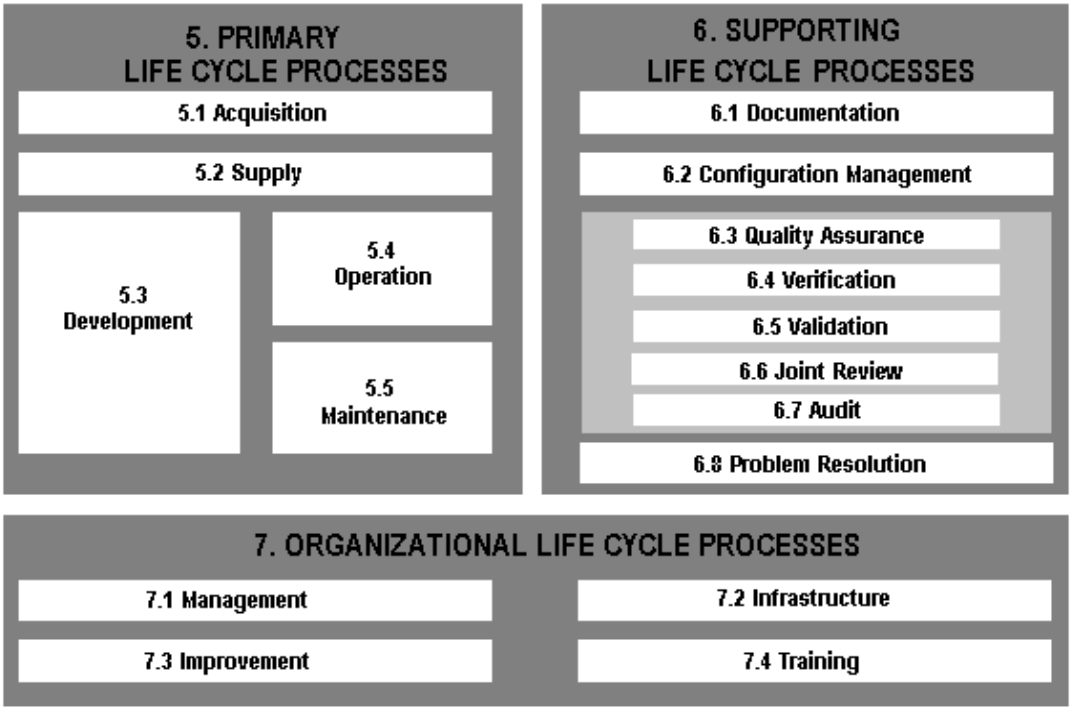

Abbildung 3-7 Die Prozesse der ISO 12207 [URL:12207]

Es existieren folgende Prozesse [URL:12207]:

- Fünf *Primary Processes*: Acquisition, Supply, Development, Operation, Maintenance. Die [ISO-12207] gliedert den *Software-Lebenszyklus* anhand von fünf verschiedenen **Rollen** (genannt: *major parties*), welche in den Lebenszyklus der Software, abhängig vom jeweils aktuellen Prozess, eingebunden sind. Diese Rollen führen einen Prozess entweder selbst durch oder aber sind verantwortlich für die Durchführung und initiieren und überwachen den Prozess.
  Acquirer: Dieser Rolle obliegt der Beschaffungsprozess (*acquisition process*) von der Bedarfsdefinition und Initiierung, Ausschreibung und Vertragsgestaltung über die Kontrolle des Auftragnehmers bis zur Abnahme.
  Supplier: Dieser Rolle obliegt der Lieferprozess (*supply process*) von der Angebotsabgabe und Vertragsunterzeichnung bis zur Auslieferung.
  Developer: Dieser Rolle obliegt der eigentliche Entwicklungsprozess (*development process*) der unter Umständen bereits die System-Anforderungsanalyse beinhaltet und bis zur Installation und Einführung reicht.
  Operator: Dieser Rolle obliegt der Betriebsprozess (*operation process*), welcher die Überprüfung, Durchführung und Unterstützung des Betriebes umfasst.
  Maintainer: Dieser Rolle obliegt der Wartungsprozess (*maintenance process*) von der Modifikation über Migration bis zur Ausmusterung.
- Acht *Supporting Processe*: Documentation, Configuration Management, Quality Assurance, Verification, Validation, Joint Review, Audit, Problem Resolution.

- Vier *Organizational Processes*: Management, Infrastructure, Improvement, Training. Dabei ist zu bemerken, dass die *Organizational Processes* von allen *Primary Processen* angesteuert werden, die *Primary Processes* iniziieren auch die *Supporting Processe*s, und sie aktivieren sich z. T. auch gegenseitig.

3.2.2 Abdeckung der Prozesse der ISO 12207 durch das V-Modell

In diesem Abschnitt wird die Abdeckung der Prozesse der [ISO-12207] durch das V-Modell vorgestellt. Die Gliederung der nachfolgenden Zuordnungen orientiert sich an der Struktur der [ISO-12207]. Diese Zuordnung zeigt, dass das V-Modell bis auf wenige Ausnahmen, die alle ausserhalb des Anwendungsbereichs des V-Modells liegen, die [ISO-12207] erfüllt - die Zuordnung bedeutet dabei nicht Gleichheit, weil die Aufteilung der ISO-Prozesse und Aktivitäten des V-Modells unterschiedlich aufgeteilt sind. Produkte werden nicht zugeordnet, weil die [ISO-12207] produktbezogene Regelungen nicht enthält.

Es werden folgende Prozesse untersucht [URL:12207]:

(1) Zuordnung von Aktivitäten des V-Modells zu den Primary Processes.

(2) Supporting Processes.

(3) Organizational Processes.

zu (1) Zuordnung von Aktivitäten des V-Modells zu den *Primary Processes*

Die Prozesse *Acquisition* und *Supply* werden vom V-Modell nur teilweise abgedeckt.

„Der Prozess *Development* kann vollständig über das V-Modell realisiert werden. Dabei werden im Development Prozess des [ISO-12207] nicht nur SE-Aktivitäten des V-Modells angesprochen, sondern auch QS-Aktivitäten und z. T. auch PM-Aktivitäten und KM-Aktivitäten.

In den *Aufgaben* der [ISO-12207] werden zu den SE-Entwicklungsaktivitäten die zugehörigen Prüftätigkeiten innerhalb des Development Prozesses mit aufgeführt. Obwohl die [ISO-12207] eigenständige Qualitätssicherungs-Prozesse definiert hat (siehe *Supporting Processes*), werden diese hier nicht referenziert. Aus diesem Grunde umfasst der *Development Process* das Submodell **SE** (ausgenommen die Hardware) und Teile des Submodells QS des V-Modells.

Der *Process Maintenance* wird insofern über das V-Modell abgedeckt als allgemeine SW-Wartung-Äktivitäten und Änderungsaktivitäten betroffen sind. Den *Aspekt Software Retirement* betrachtet das V-Modell jedoch nicht.

Nachdem das V-Modell ein Entwicklungsstandard ist, wird der Prozess *Operation* nicht geregelt, jedoch sind einige QS-, KM- und PM-Aktivitäten durchaus auch auf den *Betrieb* übertragbar *Acquisition Processes.*

zu (2) Supporting Processes

Das V-Modell kennt keine separate *Dokumentationsaktivität*. Dokumentation ist grundsätzlich in der Aktivitätendurchführung mit enthalten. Eine Zuordnung der V-Modell-Aktivitäten

ist aus diesem Grund nicht möglich, wenngleich der Dokumentationsprozess gesehen über das gesamte V-Modell selbstverständlich vollständig erfüllt wird.

Der *Configuration Management Process* der [ISO-12207] behandelt neben der Konfigurationsverwaltung und dem Änderungsmanagement des V-Modells auch *Configuration Evaluation*. Auf Seiten des V-Modells kann hier keine unmittelbare Parallele gefunden werden.

Nach [ISO-12207] werden *Joint Review*, *Audit*, *Verification* und *Validation* als Techniken des *Processes Quality Assurance* angesehen. Nachdem jedoch der *Quality Assurance Process* nur grob angerissen wird, fehlt eine explizite Einordnung dieser Techniken. Wenngleich das V-Modell alle die aufgeführten QA-Aspekte abdeckt, ist aufgrund der in [ISO-12207] allgemein gehaltenen Aussagen und der Mischung zwischen Aktivitäten (hier: tasks) und Methoden (hier: techniques) eine 1:1-Zuordnung zwischen V-Modell und [ISO-12207] nicht immer möglich.

Das V-Modell trennt strikt zwischen den Aktivitäten: *Prozess-Prüfung*, *Produktprüfung* und *Durchführungsentscheidung,* sowie auf Seiten der Methodenzuordnung: Zwischen *Audit*, *Review* (technischer Review und Durchführungsreview) und *Testen*, wobei sich der Einsatz der jeweiligen Methode an den "Prüfobjekten" (Software, Produkt, Aktivität, Projektverlauf) orientiert.

zu (3) Organizational Processes

Die beiden Prozesse *Management* und *Infrastructure* sind im V-Modell durch das Submodell **PM**, abgedeckt.

Ein *Improvement-Process* als solcher findet sich im V-Modell nicht, jedoch kann der Weg über Prozessprüfungen und das Hilfsmittel Projekthistorie beschritten werden, wodurch sich Ergebnisse für den Improvement Process ableiten liessen.

Für den *Training-Process* sind im V-Modell die Schulung/Einarbeitung und Einweisung der Mitarbeiter enthalten“ [URL:12207].

3.3 Eigenschaften des V-Modells

Das VM hat sehr viele Vorteile, die man bei der SW-Entwicklung nutzen kann. Dieser Standard wurde deswegen für die Diplomarbeit ausgewählt, weil er zur Erreichung folgender Ziele beiträgt, die für das UE-Prozess-Modell, das im Kapitel 4 dargestellt wird, u. a. von entschiedener Bedeutung sind:

- Vorteile beim VM-Einsatz in Projekten:
  - Entwicklungskosten-Ersparnis
  - Optimierung des Entwicklungsablaufs
  - Transparenz der Arbeitsabläufe
  - Weniger Abstimmungsprobleme
  - Verbesserung des logischen und strukturierten Vorgehens
  - Vorgaben bei Dokumententypen und -inhalt
  - Detaillierte und präzise Beschreibung der organisatorischen Schnittstellen
  - Klare Vergabe der Verantwortungen und Kompetenzen
  - Grosses Erfahrungspotential vergangener Projekteinsätze
- Beitrag zur Qualität:

- durch Konformität zu [ISO-12207];
- durch einheitliche Produktbeschreibung wird die Lesbarkeit der Produkte und die Durchführung von Prüfmassnahmen erleichtert;
- durch standardisiertes Vorgehen wird die Vollständigkeit der zu liefernden Ergebnisse am ehesten gewährleistet;
- durch definierte Zwischenergebnisse werden frühzeitige Prüfmassnahmen möglich;
- durch Erfüllung von QS-Vorschriften: Das VM enthält detaillierte Regelungen für den gesamten Prozess der IT-Systemerstellung einschliesslich der Tätigkeitsbereiche Projektmanagement, Qualitätssicherung und Konfigurationsmanagement. Die Mindestanforderungen der entsprechenden technischen Anforderungen nach [ISO- 900x] werden erfüllt.

- Vorteile beim standardisierten Vorgehen:
  - macht die Kalkulation des Aufwandes transparenter. Kostenrisiken werden besser erkennbar;
  - ermöglicht die Reduzierung des Ressourcen-Einsatzes;
  - universelle Lösungsansätze werden erkennbar und damit mehrfach verwendbar;
  - Fehlentwicklungen werden früher erkannt;
  - Die Schulungskosten werden verringert.

- Verbesserung der Kommunikation zwischen allen Beteiligten:
  - Reduktion der Missverständnisse zwischen den Beteiligten durch die Verwendung definierter, standardisierter Begriffe;
  - Der Entwickler und der Anwender werden bei der Formulierung ihrer Forderungen/Beschreibung ihrer Anteile oder Ergebnisse unterstützt;
  - Vereinfachung bei der Einarbeitung anderer Beteiligter (Personal), anderer Firmen.

- Erfüllung von Sicherheitskriterien:
  - „Systeme und Software, die durch unabhängige und neutrale Organisationen evaluiert und zertifiziert werden sollen, müssen die Anforderungen der zugrundeliegenden Sicherheitskriterien erfüllen.
  - Die Regelungen des VM erfüllen die Anforderungen des zugrundeliegenden Kriterienkatalogs zur Sicherheit hinsichtlich des Entwicklungsvorgehens und der zu erstellenden Produkte, und alle dort enthaltenen Anforderungen können im Rahmen der Vorgehensweise nach dem VM festgelegt und kontrolliert realisiert werden.
  - Unter IT-*Sicherheit* wird im VM sowohl Sicherheit im Sinne von "Security" (klassische IT-Sicherheit) als auch im Sinne von "Safety" (Verfahrens- oder Betriebssicherheit) verstanden„ [URL:12207].

3.4 Fazit

Neben den vielen Vorteilen weist VM auch einige Defizite auf. Das vordergründigste liegt in der Beanspruchung des VM, ein Standard (was es zum grössten Teil auch ist) zu sein, ohne von den relevanten ISO-Normen, wie [ISO-13407], [ISO-15504], [ISO-8402], [ISO-12119], [ISO-9241] Notiz zu nehmen (ausser ISO 12207, 900x). Obwohl Normen in Hinblick auf die

Vereinheitlichung nicht nur der Standardisierung behilflich sind, sondern sie legen einen entscheidenden Grundstein für eine internationale Akzeptanz.

Im VM gibt es ein Problem mit der Begriffssemantik, wo die Erfüllung der Qualität nur einseitig betrachtet wird, indem man nur die Vollständigkeit und Korrektheit achtet. Zu Betrachtung des Qualitätsbegriffes muss man die Definition der [ISO-8402] praxisbezogen hinzuziehen. Die Norm besagt: „Gesamtheit von Merkmalen einer Einheit bzgl. ihrer Eignung, festgelegte und vorausgesetzte Erfordernisse zu erfüllen“. Eine zentrale Bedeutung hat also die Zuordnung des Begriffspaares *Anforderung* und *Merkmal*, denn nur wenn geklärt wird, was die Tätigkeits-Anforderung ist und welches Merkmal ein SWP für die Erfüllung der Tätigkeit haben muss, kann SW-Qualität gemessen werden. Im VM wird zwar auf die ISO 9000 verwiesen und die Qualitätsdefinition der [ISO-8402] geltend gemacht, aber es wird kein Vorgehen beschrieben, wie mit diesen Paaren von Anforderungen und Merkmalen umzugehen ist. Ebenso setzt man sich mit den Begriffen der *Gebrauchstauglichkeit* [ISO-9241] und der *Benutzbarkeit* [ISO-12119] im VM [Drös´00] in keinem Kapitel auseinander.

Der Grundfrage: „Wozu soll ein SWP nach dem SW-Entwicklungsprozess des VM entwickelt werden?“ wird nirgendwo nachgegangen - da VM verlangt nicht nach der Beantwortung der Frage der Produkt-Nutzung. Es gibt keine Untersuchungen darüber, zu welchen Zwecken der Kunde (Anwender oder Benutzer) das SWP nutzen kann, welches er mit Hilfe des VM entwickeln lässt?

Im VM wird demnach nur die system-technische Qualität geprüft, nicht aber die Nutzungs-Qualität, weil man nur die system-technische Anforderungen und deren Prüfkriterien analysiert und berücksichtigt. Es sei nur am Rande bemerkt, dass `der Begriff „Merkmal“ im gesamten Regelungsteil des Submodells` **`QS`** `überhaupt nicht mehr vor kommt, obwohl er noch im` **`SE1`** `genannt wurde.` Denn dabei stellen sich die entscheidenden Fragen: Was sind eigentlich die *fachlichen Anwenderforderungen*, die aus **SE1** kommen, obwohl keine Nutzungs-Anforderungen und deren Prüfkriterien entwickelt worden sind, um die Nutzungs-Qualität zu sichern? Welche Massnahmen ergreift man zur Ermittlung der *Externen Vorgaben (AG)* in **SE1**, um dort das gegebene Nutzungsziel mit einem künftigen SWP erreichen zu können? (Gemeint ist, hier einen Soll-Zustand zu analysieren.)

Problematisch ist besonders die Auffassung zur Validierung von Anforderungen (mit Benutzerbeteiligung), weil z. B. im **SE1.7** die Anwenderforderungen lediglich auf Plausibilität untersucht werden.

Die Erfüllung fachlicher Anforderungen im VM reicht für die Erzielung der Nutzungs-Qualität nicht aus. Mit Hilfe von UE (siehe Kapitel 4) wird erheblich mehr zur Entwicklung und Sicherung von Nutzungs-Qualität geleistet.

Die Aktivität **SE9** *Überleitung in die Nutzung* „umfasst Tätigkeiten, die notwendig sind, um ein fertiggestelltes System an der vorgesehenen Einsatzstelle zu installieren und in Betrieb zu nehmen...“ [URL:V-Modell]. Im Kapitel 4 der Diplomarbeit wird genauer auf das Problem des Projektabschlusses eingegangen. Hier sei nur bemerkt, dass mit der Installation des SWP kaum an eine Beendigung des SW-Entwicklungs-Projektes gedacht werden darf. Aus der Sicht der Nutzung muss Pflege- und Wartung des SWP betrieben werden, und zwar nicht nur aus sw-technischer Sicht, sondern vor allem im Sinne des Usability-Engineering (dazu siehe unter *SW-Einführungs-Projekt* **(UE-Pfl)**).

Zusammenfassend kann man sagen, dass das VM die Forderungen in [ISO-13407] (genauer Punkt 7 der Norm) bezüglich der benutzerorientierten Gestaltung interaktiver Systeme aus den o. g. Gründen (aus der Sicht der Benutzerbeteiligung) kaum erfüllen könne (Vergleich mit Kapitel 2.3 der Diplomarbeit). Es ist möglich gebrauchstaugliche SWP mit dem VM zu entwickeln, strenggenommen aber nur dann, wenn systematisch Methoden des Usability-Engineering in das VM einbezogen werden (siehe Kapitel 4). Dies ist eine notwendige Voraussetzung für die Sicherung der Gebrauchstauglichkeit der SWP.

Der Vergleich sollte deutlich machen, welche Aspekte des Usability-Engineering und der ISO Normen im V-Modell bis jetzt nicht genügend berücksichtigt worden sind, was allerdings nicht ausschliesst, sie künftig in das VM einbeziehen zu können.

4 Ein Prozess-Modell für Usability-Engineering

Die “prozessorientierte Produkt-Entwicklung“ ist das grosse Erbe der Ingenieurs-Wissenschaften an die heutige SW-Entwicklung [VDI-2221]. Denn das ist der Ansatz, den man immer wieder aufsuchen muss, wenn man nach der Qualität des (künftigen) Produktes fragt. Das Produkt selber ist nur ein Ergebnis von Prozessen, demnach spielen diese die entscheidende Rolle bei der Qualität des Produktes: Während der Prozess-Abläufe beginnt man mit der Sicherung der Qualität.

Der prozessorientierten Produkt-Entwicklung liegt ein **Prozess-Modell** zugrunde, das folgendermassen definiert werden kann [Winz'00]:

„Eine Tätigkeit oder Operation, die Eingaben erhält und diese in Ergebnisse umwandelt, kann als Prozess angesehen werden. Fast alle Tätigkeiten und Operationen im Zusammenhang mit einem Produkt sind Prozesse“.

Es gibt verschiedene Ebenen von Prozessen. Das Entscheidende daran ist, wie man sinnvoll die zahlreichen Prozesse definiert und untereinander verknüpft, damit die Herstellerorganisation funktionieren kann - das Output des eines Prozesses bildet den direkte Input für den nächsten. Die systematische Erkennung und Steuerung dieser verschiedenen Prozesse innerhalb einer Organisation und vor allem die Wechselwirkung zwischen ihnen, machen den *prozessorientierten Ansatz* bei der Produkt-Entwicklung aus.

Eine prozessorientierte Vorgehensweise bei der SW-Entwicklung und ihre Dokumentation bieten Vorteile für die Vorbereitung der Zertifizierung von SWP. Die Unternehmen können auf der Basis ihrer Prozesse den Nachweis der Einhaltung der Normen erbringen.

Die Motivation für den Entwurf eines UE-Prozess-Modells liegt darin, ein Vorgehens-Modell zur Sicherung der Gebrauchstauglichkeit bei der Entwicklung von Software zu erstellen. Denn die Aufgabe, ein gebrauchstaugliches SWP (Qualität) zu produzieren, und zwar nicht nur einmalig, sondern immer wieder, bedarf besonderer Prozess-Eigenschaften.

Ergebnis dieses Kapitels ist ein Prozess-Modell für UE, ergänzt um Hinweise für die (exemplarische) Umsetzung in einem Software-Entwicklungsprojekt. Es soll die bei der Software-Entwicklung einzuhaltenden Arbeitsschritte beschreiben, soweit sie für die Sicherstellung der Gebrauchstauglichkeit wichtig sind. Dazu wurde das V-Modell um Aspekte des UE erweitert und parallel dazu das Prozess-Modell erstellt (weil man UE nicht ohne SE betreiben kann), was auch ein Ergebnis der Diplomarbeit ist.

Ein Projekt zur Software-Entwicklung muss von Anfang an die Aspekte des Usability- und Software-Engineering berücksichtigen. Solch ein Modell für ein **SW-Entwicklungs-Projekt** wird dann unterteilt in vier Unter-Projekte : *SW-Planungs-Projekt* (Usability-Engineering), *SW-Entwurfs-Projekt* (SW-Entwurf), *SW-Implementierungs-Projekt* (Programmentwicklung) und *SW-Einführungs-Projekt* (Installation, Wartung und Pflege) (siehe Abbildung 4-1). Diese Aufteilung ist sehr hilfreich bei der vertraglichen Abstimmung des Projekts. Dadurch werden Verträge mit dem Anwender möglich, die den SE- und UE-Prozess für den Hersteller kalkulierbarer werden lassen. Ausserdem erleichtern solche Vertrags-Module, die Konfliktpotentiale rechtzeitig aufzudecken und zu beheben. Wenn ein Projekt bereits in der Planung scheitert, ist der Schaden begrenzt.

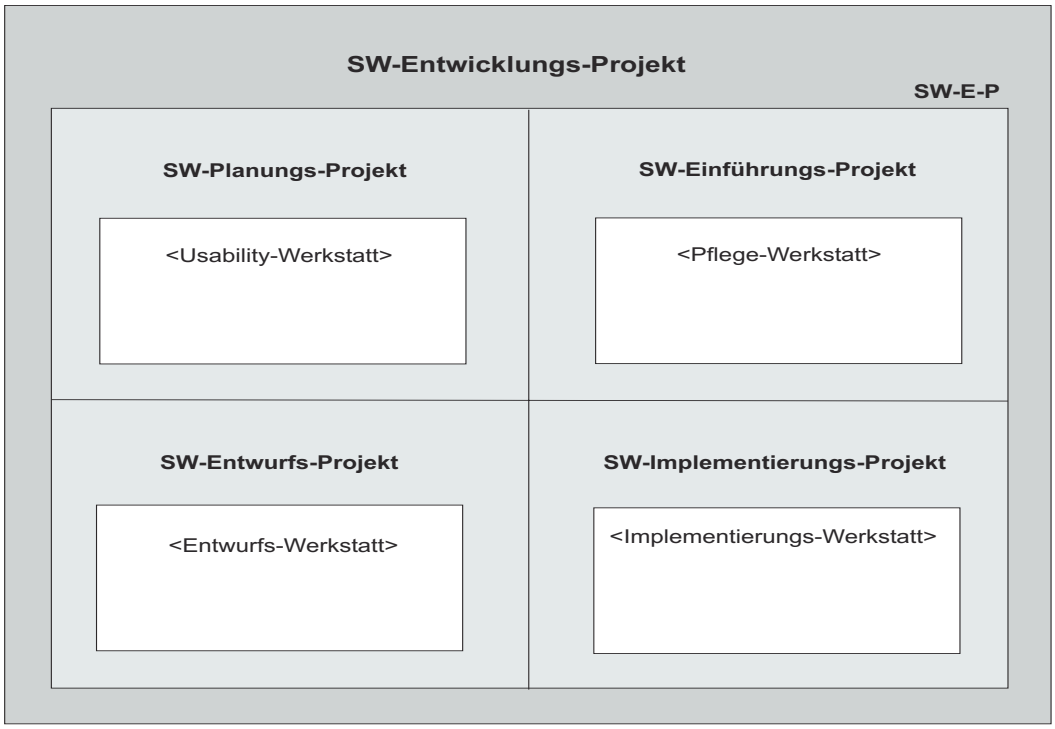

Abbildung 4-1 SW-Entwicklungs-Projekt

An dieser Stelle sei ausdrücklich darauf hingewiesen, dass es sich bei einem SW-Entwicklungs-Vertrag weder um einen Kauf-, noch um einen Werk-Vertrag handelt. Im Unterschied zu einem Werk-Vertrag, in dem nur das Ergebnis einer Leistung (z. B. "es funktioniert") vereinbart wird, kommt es beim SW-Entwicklungs-Vertrag zusätzlich auf eine der wichtigsten Vereinbarungen an, nämlich über den Prozess der SW-Entwicklung. Dieser besteht aus vielen Aufgaben des Herstellers, die ohne Mitwirkung des Anwenders nicht effektiv bewältigt werden können [Müll/West'94]. Die Mitwirkungspflicht des Anwenders ergibt sich aus dem Umstand, dass die Leistung des Softwareproduktes erst mittels einer Analyse- und Entwurfs-Phase spezifiziert werden kann.

Aus der Praxis der SW-Entwicklung kann man entnehmen, dass die Anforderungen einem *Entwicklungs- und Änderungs-Prozess* unterliegen. Während der Anwender immer besser versteht, wie das angebotene technische Potential in Leistungsanforderungen umgesetzt werden kann, versteht der Hersteller immer besser die Nutzungs-Anforderungen der Anwenderorganisation. Ausserdem endet aus Sicht des UE ein Projekt zur SW-Entwicklung nicht mit der Abnahme des Produkts durch den Anwender, sondern erst nach dem zweiten "Release". Das hier vorgestellte Modell zur SW-Entwicklung orientiert sich nicht mehr am klassischen Phasen-Modell, sondern an einem Zyklus, der zumindest die erste Nutzungs-Phase im realen Nutzungs-Kontext einbezieht, weil das SWP erst in der kontextabhängigen Nutzungs-Phase (in SW-Einführungs-Projekt) fertiggestellt werden kann (sog. **design-use cycle**, sieh Abbildung 4-2).

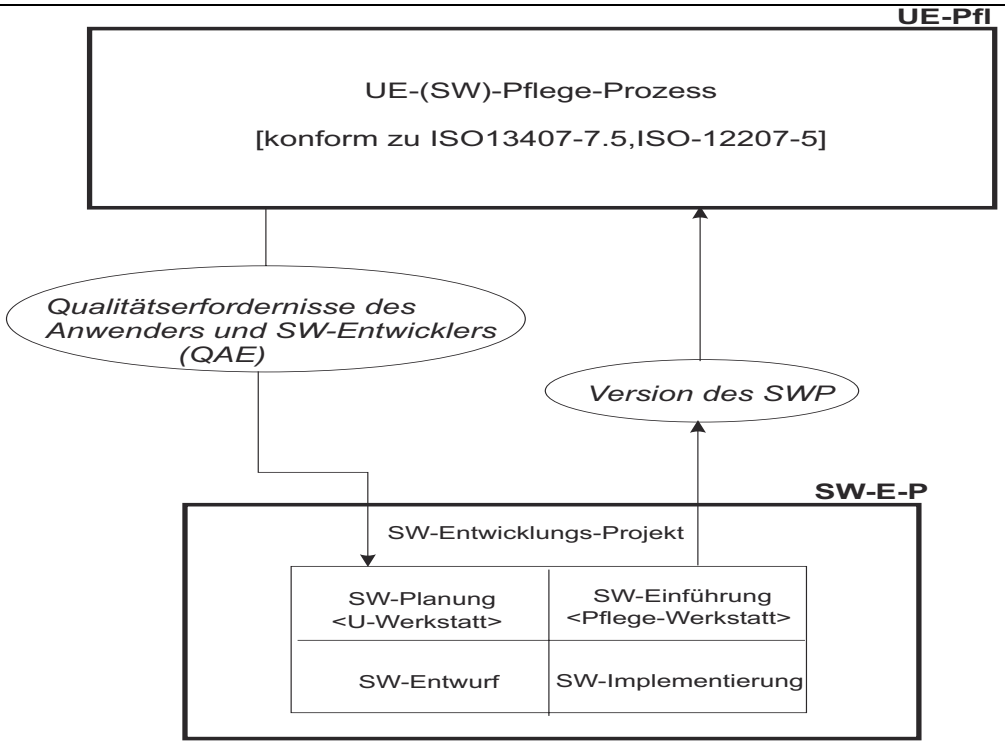

Abbildung 4-2 UE-(SW)-Pflege-Prozess

Der Anwender soll beim Vertragsabschluss eine Regelung der notwendigen Änderungen verlangen, die sich nicht nur aus dem Prozess der Anforderungs-Entwicklung und dem Prototyping ergeben, sondern auch aus der Aufdeckung der Nutzungs-Probleme im realen Nutzungs-Kontext. Die Vorgehensmodelle vieler Anwender enthalten organisierte Änderungs-Prozesse auf der Grundlage von Fehlermeldungen (Wartung). Diese Vorgehensmodelle sollten vertraglich mit dem Wartungs- und Pflege-Prozess des Herstellers abgestimmt werden, damit die Möglichkeit gesichert ist, dass auch nach der Einführung einer Software die Entwicklung von Usability-Anforderungen im vorgesehenen Nutzungs-Kontext auf Grund aufgedeckter Nutzungsprobleme möglich ist [Dzid/Frei´00].

Aus Kostengründen ist die vertragliche Vereinbarung der Änderungen während der Anforderungs-Entwicklung und beim Prototyping am wichtigsten; denn die Kosten für Änderungen in der Pflege(Nutzungs-)Phase sind etwa zehn mal höher als in der Implementierungs-Phase und sechzig mal höher als in der Entwurfs-Phase [Land´95]. Die Kosten der Änderungen für festgestellte Abweichungen von Mindestanforderungen sollte vertraglich vereinbart der Hersteller tragen, weil nur er in der Lage ist, Normen (u.a. der Reihe ISO 9000, ISO 9241, ISO 13407, ISO 12027) anzuwenden. Norm-Konformität liegt somit eindeutig im Verantwortungsbereich des Herstellers [Dzid/Frei´00].

4.1 Der Dialektische Prozess

In diesem Kapitel sollen die dialektischen Aspekte des Problemlösens als Informationsverarbeitung vorgestellt werden, weil sie eine wesentliche Rolle in dem kommenden UE-Prozess-Modell (Kapitel 4.2) spielen werden, um die Anforderungs-Definition entwickeln zu können.

Genauergesagt geht es hier um das schrittweise Problemlösen nach Dörner [Dörn´79], wo man Dialektische und Synthese Probleme betrachtet. Hiermit werden die in Kapiteln 2 und 4.2 referierte Modelle der Anforderungs-Entwicklung inhaltlich vertieft.

Problem

Zunächst ist zu klären, was ist ein Problem: „Ein Individuum steht einem Problem gegenüber, wenn es sich in einem inneren oder äusseren Zustand befindet, den es aus irgendwelchen Gründen nicht für wünschenswert hält, aber im Moment nicht über die Mittel verfügt, um den unerwünschten Zustand in den wünschenswerten Zielzustand zu überführen“ [Dörn´79]. Daraus ergibt sich, dass ein Problem aus drei Komponenten besteht:

- unerwünschter Anfangszustand;
- erwünschter Zielzustand;
- *Barriere*, die die Transformation vom Anfangszustand zum Zielzustand im Moment verhindert.

Vollständigkeitshalber ist dem hinzuzufügen, dass man zwischen Problemen und Aufgaben („geistige Anforderungen, für deren Bewältigung Methoden bekannt sind“) unterscheiden muss: „Aufgaben erfordern ein reproduktives Denken, beim Problemlösen aber muss etwas Neues geschaffen werden.“

Typen von Barrieren

Barrieren können von verschiedener Art sein, die verschiedene Formen der Überwindung/ des Problemlösens verlangen. Dabei spielt die Art des Realitätsbereiches (Sicherheit der Kausalität, Eigendynamik, Vorhandensein, Komplexität) auch eine Rolle.

Die *Überwindung* einer Barriere (Lösungsfindung) besteht darin, dass man eine Anzahl von Operationen findet, die die Transformation des unerwünschten Anfangszustands in den erwünschten Zielzustand ermöglichen.

Die *Überwindung einer Synthese-Barriere* besteht darin, die richtigen Operationen (Mittel) überhaupt erst zu finden, und sie dann in eine richtige Kombinationsfolge zu bringen. Die Hauptaufgabe liegt in einer Zusammenstellung (Synthese) einer brauchbaren Menge an Operationen. Das Hauptproblem liegt aber in der sog. Voreinstellung (gelernte Einstellungen und Denkgewohnheiten), die die Hauptursache dafür ist, dass viele Probleme für Individuen Synthese-Barrieren beinhalten.

In vielen Fällen weiss man nur, dass eine Situation verändert werden muss, aber der Zielzustand kann noch nicht formuliert werden. Bei diesem Problem muss man also „dialektisch“ vorgehen und zwar nach der folgenden Formel:

Man macht einen Vorschlag/Entwurf, diesen Vorschlag/Entwurf wird man auf Widersprüche prüfen, die Widersprüche wird man dann beseitigen, und dann den Vorschlag/Entwurf erneut prüfen.

Die oben geschilderte Formel trägt zur Lösung der *Dialektischen Barriere* bei, die meist in einem **Dialektischen Prozess** überwunden wird, indem ein Vorschlag/Entwurf für den Zielzustand auf Widersprüche geprüft wird und dann entsprechend verändert wird.

Je unklarer ein Problem hinsichtlich des Zielzustandes ist, in um so höherem Masse muss es sich dem Dialektischen Prozess unterwerfen, und desto mehr muss man also versuchen, durch Erzeugung und nachfolgende Beseitigung von Widersprüchen, sukzessiv eine Präzisierung der Zielvorstellungen zu erlangen.

Im Gegensatz dazu, wenn man in einer Problemsituation schon den Zielzustand und die Mittel kennt, mit Hilfe derer man den angestrebten Zielzustand erreichen kann, dann liegt das Problem nur in der richtigen Kombination der Mittel – Überwindung der *Interpolations-Barriere*.

Im Falle, wenn man den Zielzustand kennt und die Mittel ganz oder teilweise unklar sind, so hat man mit der *Synthese-Barriere* zu tun. Und wie schon gesagt, wenn man nicht weiss, was man eigentlich will, so muss man erst die *Dialektische Barriere* überwinden.

Zur Veranschaulichung eine Tabelle (Abbildung 4-3) der Klassifizierung von Barrieren nach den Dimensionen „Klarheit des Zielzustandes“ und „Bekanntheitsgrad der Mittel“ :

		Klarheit des Zielzustandes	
		hoch	**gering**
Bekanntheitsgrad der Mittel	**hoch**	*Interpolations-Barriere*	*Dialektische Barriere*
	gering	*Synthese-Barriere*	*Dialektische Barriere und Synthese-Barriere*

Abbildung 4-3 Klassifizierung von Barrieren [Dörn´79]

Der Weg, den man anstrebt, führt also von der *Dialektischen Barriere* über *Synthese-Barriere* zur *Interpolations-Barriere*, um so eine hohe Klarheit des Zielzustandes und hohen Bekanntheitsgrad der Mittel zu erreichen. Denn so kann man sich sicher sein, dass man alle Barrieren überwunden hat und eine neu Lösung für ein Problem gefunden hat, d.h. Innovatives erschaffen hat.

Diese Gedanken des Problemlösens und die oben beschriebene Vorgehensweise werden in dieser Diplomarbeit für des UE-Prozess-Modell genutzt, um das Problem der Entwicklung des SWP lösen zu können, und die Nutzungs-Qualität zu sichern.

Die Gegenüberstellung des Nutzungskonzeptes einerseits und des Systemkonzeptes andererseits provoziert den Widerspruch, der den Synthese-Prozess anstösst (aus dem Prinzip von These und Antithese entsteht als ein neues Produkt die Synthese).

Zusammenfassend kann man sagen, dass man ohne diesen Dialektischen und Synthese-Prozess nicht zur Innovation kommt (ähnlich wie: „ohne den Widerspruch von Feuer und Wasser gibt es keinen Dampf „).

4.2 Das UE-Prozess-Modell

Vor Entwicklung und Erstellung des UE-Prozess-Modells wird zunächst eine Ausarbeitung der Bestandteile (Aktivitäten und Produkte) des UE-Prozesses vorgenommen. Das Ergebnis des UE-Prozesses ist die Anforderungs-Definition, die für den SW-Entwurf bereit zu stellen ist. Der Prozess besteht u. a. aus einer bestimmten Abfolge von Arbeitsschritten, wie die Erstellung von Kontext-Szenarien, Ableitung von Anforderungen, Erstellung von Use-Szenarien und Use-Cases, das Usability-Prototyping und die Ableitung von Prüfkriterien. Entscheidend für das Prozess-Modell ist, *wie* diese Arbeitsschritte sinnvoll durchzuführen sind, um eine aussagekräftige Anforderungs-Definition zu erhalten.

Der Formalismus des UE-Prozess-Modells

Das Prozess-Modell für UE wird in der für den Regelungsteil des V-Modells typischen Form erstellt und dokumentiert (V-Modell'97, [Drös´97]). Die strukturierte Darstellung des Vorgehens ist nicht nur logisch nachzuvollziehen, sondern weit verbreitet und anerkannt. Dieser V-Modell-Formalismus wurde für die Beschreibung des Usability genutzt, um ein UE-Prozess-Modell zu entwickeln, und dann das V-Modell um das UE-Prozess-Modell zu erweitern. Die Grundelemente des UE-Prozess-Modells sind Aktivitäten und Produkte, die während des SW-Entwicklungs-Prozesses durchgeführt bzw. bearbeitet werden. Der Zusammenhang zwischen Aktivitäten und Produkten ist durch den Produktfluss dargestellt. Sowohl die genaue Beschreibung des V-Modell-Formalismus, als auch die Generierung des Produktflusses wurde schon im Kapitel "Das V-Modell" vorgestellt.

Das UE-(Planungs-)Projekt

Wie schon im vorigen Abschnitt deutlich gemacht worden ist, orientieren sich die SW-Entwicklungs-Projekte nicht mehr am Werk-Vertrag, sondern am Vertragsmodell für Bau- und Anlagen-Projekte (siehe [Müll/West'94], Seite 56). Analog zur Vorgehensweise des Planungs-Projektes, z. B. im Architekturbüro oder in einer "Planungs-Abteilung", fängt man bei der Software-Entwicklung in der "**Usability-Abteilung**" mit der *Anforderungs-Entwicklung* an. In dieser "UE-Abteilung" arbeitet ein **Usability-Team**, bestehend einerseits aus *U-Analytiker*, *Usability-Engineer* und *U-Prüfer* mit dem Benutzer/Anwender und andererseits mit dem **IT-Team** eng zusammen. Die Teams entwickeln gemeinsam eine Anforderungs-Definition für das zu entwickelnde Softwareprodukt, welches gebrauchstauglich und normkonform sein soll. Das gesamte Team arbeitet nach einem UE-Prozess-Modell, das nachfolgend vorgestellt wird.

Es sei hier angemerkt, dass diese Art der Vorbereitung von SW-Entwicklungs-Projekten zwar wünschenswert ist, aber noch kaum Eingang in die tägliche Praxis gefunden hat.

Die Sichten des UE-(Planungs-)Projektes

Ein UE-Prozess besteht aus drei Sichten: Usability-Sicht, Anforderungs-Entwicklungs-Sicht und Software-Engineering-Sicht (dazu siehe Abbildung 4-4).

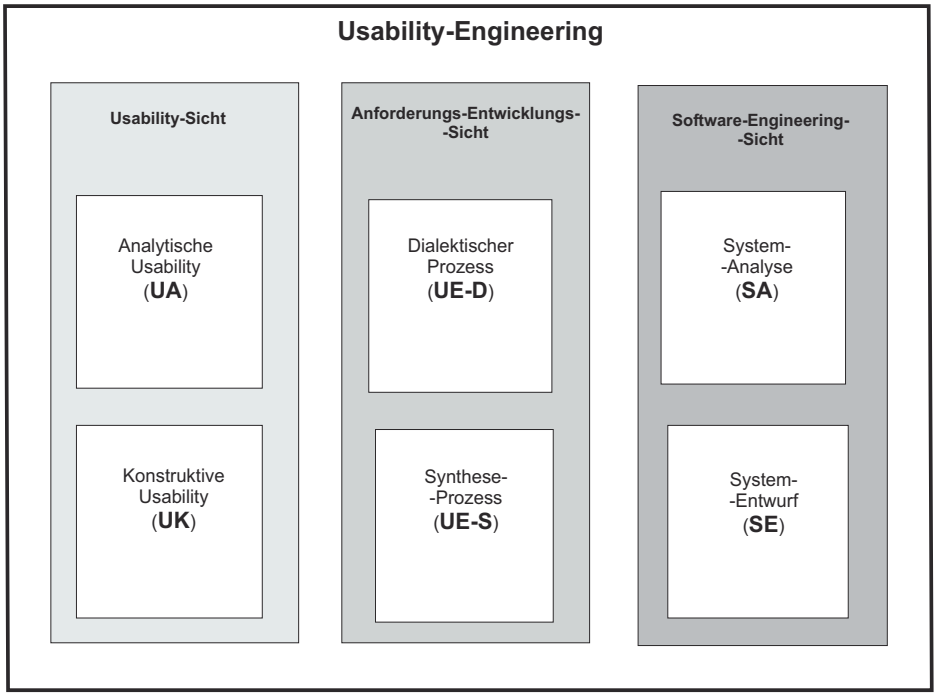

Abbildung 4-4 Sichten des UE-(Planungs-)Projektes

Die Usability- und Anforderungs-Entwicklungs-Sicht kann man als eine "Usability-Werkstatt" (im SW-Planungs-Projekt) betrachten, in der die Vorbereitungen für das SW-Entwurfs-Projekt stattfinden. Die Arbeits-Prozesse der Benutzer und Anwenderorganisatoren sowie die dafür gewünschte Nutzungs-Qualität des SWP werden in der Anforderungs-Entwicklungs-Sicht vorbereitet. Denn dort findet der Informationsaustausch zwischen dem Usability-Team und dem IT-Team statt, um die Anforderungs-Definition gemeinsam zu entwickeln. Dieser Prozess wird von einem Usability-Engineer moderiert.

Der Prozess des UE

In den folgenden Unterkapiteln werden die drei angesprochenen Sichten als **drei** (Haupt-) **Aktivitäten** des gesamten UE-Prozess-Modells betrachtet (siehe Abbildung 4-5). Sie werden gemeinsam mit ihren Aktivitäten und Produkten detailliert vorgestellt. Die Prozess-Darstellung ist, an den V-Modell-Formalismus angelehnt. Das Prozess-Modell selbst ist logisch vom Produktfluss her aufgebaut. In der praktischen Handhabung wird die Durchfüh-

rung des Prozesses nicht immer so streng betrieben werden können, d. h. je nachdem was die spezifischen Vorgaben des Kontextes oder der Nutzung diktieren, sollen Wiederholungen oder gleichzeitige Durchführung der Aktivitäten möglich sein (es kann u. U. passieren, dass keine scharfe Trennung bei bestimmten Aktivitäten zu erkennen ist, weil abschnittsweise ein fliessender Übergang verlangt wird), um Zwischenergebnisse zu optimieren oder zu validieren – allerdings muss dabei der logische Produktfluss erhalten bleiben! Der Prozess soll in erster Linie Massnahmen und ihre logischen Verknüpfungen aufzeichnen, die man unternehmen muss, um UE betreiben zu können. Bei einigen hier genannten Aktivitäten und Produkten, die schon in vorigen Kapiteln ausführlich beschrieben wurden, wird ein Hinweis stehen, im welchen (Unter-) Kapitel man sie finden kann. Die Konformität dieses Prozess-Modells für UE zu ISO Normen ist im Unterkapitel 4.3 beschrieben.

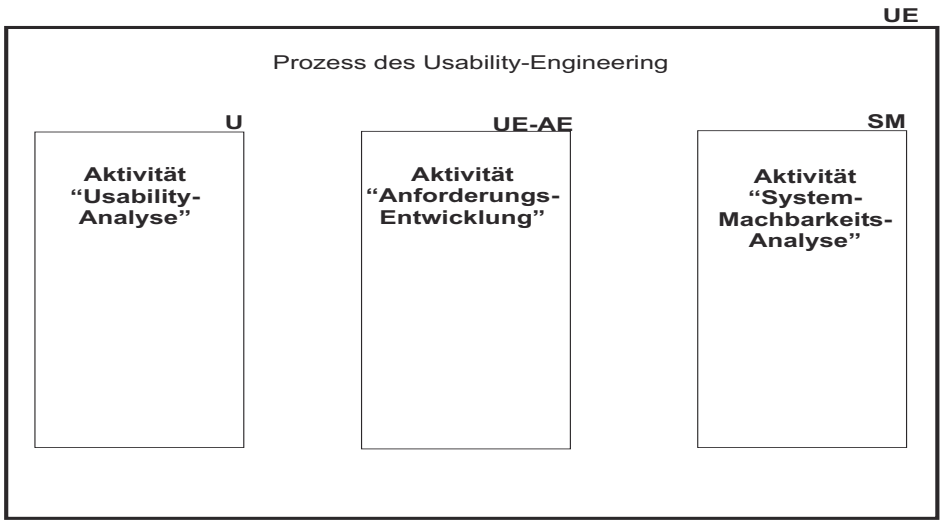

Abbildung 4-5 UE Hauptaktivitäten

Die entscheidende Aussage der beiden letzten Abbildung 4-4 und 4-5 ist nicht nur die Positionierung der drei Aktivitäten innerhalb des UE-Prozesses, sondern dass es kein **UE** für die SW-Entwicklung gibt ohne den entsprechenden Teil des SW-Engineering, der um **U** und **UE-AE** ergänzt werden muss – und umgekehrt - um gebrauchstaugliche, ISO-konforme SWP entwickeln zu können. Was in den folgenden Unterkapiteln auch ausführlich beschrieben wird.

4.2.1 Die Usability-Analyse Aktivität

Die (Haupt-)Aktivität Usability-Analyse **(U)** beinhaltet die Analytische Usability **(UA)** und die Konstruktive Usability **(UK)** (siehe Abbildung 4-6).

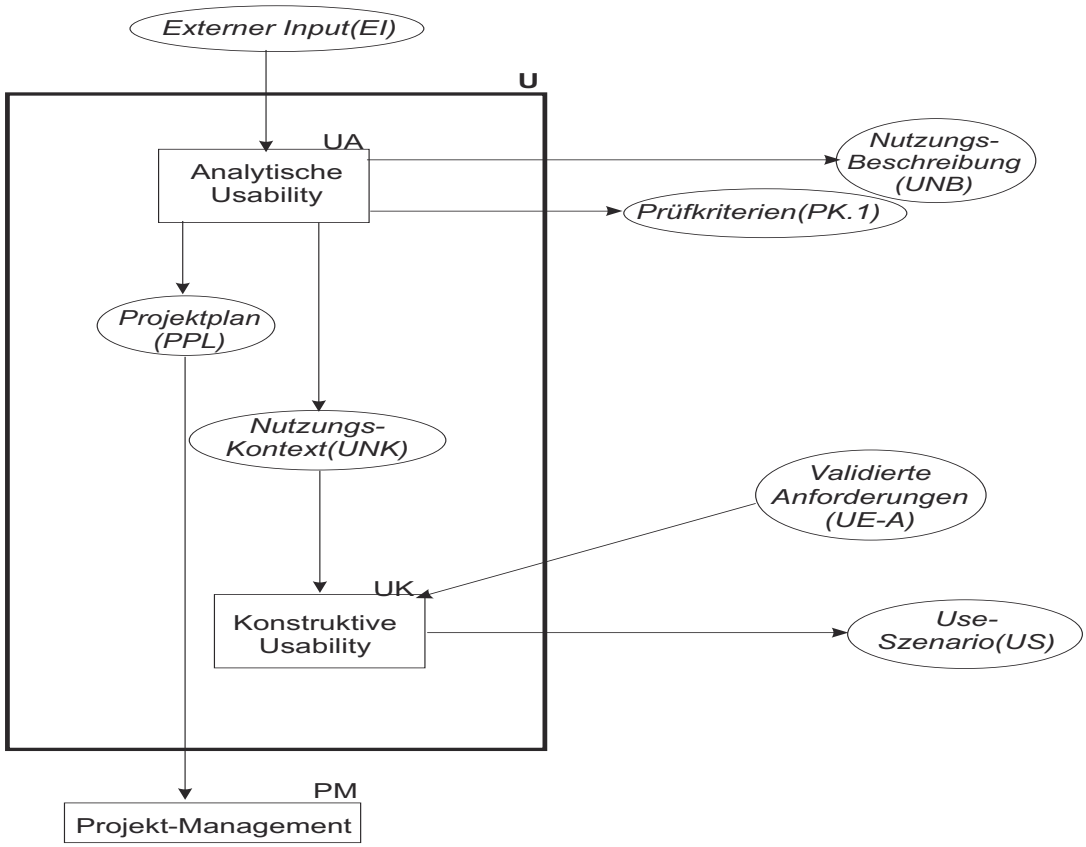

Abbildung 4-6 Usabilityanalyse-(Haupt-)Aktivität

Der Input heisst hier *Externer Input* **(*EI*)** und ist eine Tätigkeitsbeschreibung im Auftrage des potentiellen Benutzers. Dies kann sich aus einer Produkt/Prozess-Idee, einem Bedürfnis nach effizienter Erledigung von Arbeitsaufgaben o. ä. ergeben. Der U-Analytiker versucht zunächst mit dem Anwender/Benutzer die Aufgabe so zu formulieren, damit sie für die Erstellung der *Kontext-Szenarien* (siehe Kapitel 2 und 5) mittels Leitfragen brauchbar ist. Dazu können Erkenntnisse, z. B. aus einer Marktanalyse, sehr hilfreich sein.

Im folgenden ein Gesamtüberblick über einige Schritte, die bei der Anforderungs-Entwicklung zu berücksichtigen sind:

a) Die Anforderungs-Entwicklung beginnt mit der Analytischen Usability **(UA)**, in der Kontext-Szenarien **(KS-E)** erstellt und ausgewertet **(KS-A)** werden: Zunächst muss man sich über die Erfordernisse (implied needs) - sich von selbst ergebende Forderungen (Voraussetzungen), die auf gegebenen Umständen beruhen (genauer siehe Anhang: Glossar) – klar werden, dann aus Erfordernissen Aufgaben-Anforderungen ableiten (sie sollten die Anforderungen der Aufgabengestaltung gemäss ISO 9241-2 berücksichtigen).

b) Aus Erfordernissen abgeleitete Anforderungen können auch durch geeignete Kontext-Merkmale erfüllt werden, d.h. diese müssen nicht unbedingt etwas mit der Systemgestaltung zu tun haben, wohl aber mit der Gestaltung des Nutzungs-Kontextes.

c) Bei unklarer Anforderungsableitung: Erst das zugrunde liegende Erfordernis klären.

d) Aus dem *Nutzungs-Kontext* ***(UNK)*** werden Anforderungen ohne Beachtung eines existierenden SWP abgeleitet.

e) Für die Kern-Aufgaben des Nutzungs-Kontextes werden *Use-Szenarien* ***(US)*** (mittels Use-Cases **(UC)**) entwickelt und aus diesen die Anforderungen an die interaktive Nutzung des SWP abgeleitet (ggf. mittels Prototyping).

Bevor man zu der folgenden Produktfluss-Tabelle übergeht, sei an dieser Stelle betont, dass die validierten Anforderungen Voraussetzung für das Prüfen sind, was sich auf die Zustände der UE-Bereiches auswirkt. Das heisst, dass die Zustandsübergänge im UE in der Regel vom „nicht validierten Zustand" zum „validierten Zustand" gehen, weil der UE-Prozess im wesentlichen ein Prozess der Validierung von Nutzungs-Anforderungen ist. Die Zustandsübergänge müssen jedoch erst in der praktischen (langzeitigen) Erprobung des UE-Prozess-Modells zusammen mit den Zuständen im V-Modell ermittelt werden. Um zu zeigen, wie schwierig es ist, Zustände zu vereinbaren, wird an dem Beispiel des Produktflusses **(U)** deutlich gemacht, welche Übergänge denkbar sind. In den weiteren Produktfluss-Tabellen werden die Spalten **von Zustand** und **nach Zustand** in dieser Diplomarbeit weggelassen.

PRODUKTFLUSS **(U)**:

von Aktivität	**von Zustand**	***Produkt***	**nach Aktivität**	**nach Zustand**
-	-	*Externer Input(EI)*	UA	validiert
UA	zu erheben	*Nutzungs-Beschreibung(UNB)*	UE-D	validiert
UA	validiert	*Nutzungs-Beschreibung(UNB)*	SM-A	-
UA	abzuleiten	*Prüfkriterien (PK.1)*	UE-D	validiert
UA	-	*Projektplan(PPL)*	PM	vorgelegt
UA	zu erheben	*Nutzungs-Kontext(UNK)*	UK	validiert
UE-D	validiert	*Validierte Anforderungen(UE-A)*	UK	-
UK	zu erheben	*Use-Szenario(US)*	UE-S	validiert

Die Analytische Usability (UA)

Die Analytische Usability beinhaltet drei Aktivitäten: Kontext-Szenarien-Erstellen **(KS-E),** Kontext-Anforderungen-Ableiten **(KS-A)** und Projekt-Plan **(PPL)** erstellen.

In der Analytischen Usability wird die Kontext-Analyse in Form eines Kontext-Szenarios erstellt. Dies geschieht mit Hilfe von Leitfragen über die Arbeitsaufgaben, Benutzer-Qualifikationen, Probleme, Visionen. Zur Erhebung von Kontext-Szenarien (siehe Kapitel 2 und 5). Die Aufgabe hier ist zum einen ein Konzept zur Sicherung der Nutzungs-Qualität, s.g. *Nutzungs-Beschreibung* ***(UNB)*** zu liefern, die dann als Input in die Aktivität UE-Dialektischer Prozess **(UE-D)** und System-Machbarkeitsanalyse **(SM-A)** eingeht; zum anderen aber einen *Nutzungs-Kontext* ***(UNK)*** zu beschreiben, der der Aufzählung von Kern-Aufgaben (strukturell) dient und Input der Konstruktiven Usability **(UK)** ist (siehe Abbildung 4-7).

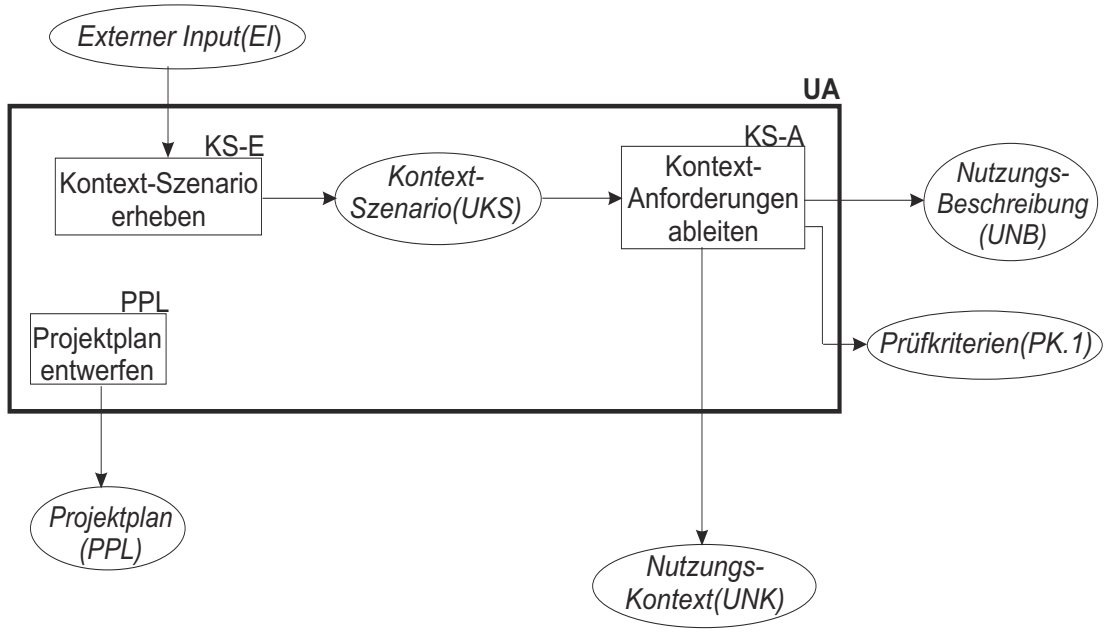

Abbildung 4-7 Die Analytische Usability (UA)

Die *Nutzungs-Beschreibung* ***(UNB)*** *(Zweck der Nutzung und Gestaltung der Nutzungsweise, sowie der Aktivitäten; u.U. die ausdrückliche Ausgrenzung von Lösungs-Alternativen – Einschränkung der Planungsvielfalt)* kann auch ein Kontext-Szenario sein, das zusätzlich Informationen über wichtige Bereiche, Randgebiete, Nachbarschaftswelten, Mittel, Dokumentationsformen, Regelwerke, Internet, virtuelle Umgebung beschreibt, die für die Nutzung eines Produkts von Bedeutung sind.

Das Kontext-Szenario wird in der Aktivität Kontext-Szenario **(KS-E)** erhoben. Die typischen Leitfragen zur Erhebung des Kontext-Szenarien sind:

- ⇒ Welche Aufgaben sollen durch die SW unterstützt werden?
- ⇒ Was könnte dadurch besser werden?
- ⇒ Was muss man gelernt haben oder wissen, um die Aufgaben zu bearbeiten? Welche Mittel helfen dabei?
- ⇒ Welche Arbeitsschritte sind durchzuführen? Wie werden Arbeitsergebnisse zurückgemeldet, gespeichert? Was soll erreicht werden?
- ⇒ Welche Störungen treten auf? Was passiert dann?
- ⇒ Welche Folgen haben Fehler? Kann aus Fehlern gelernt werden?

Bei der Durchführung der Erhebung ist auf folgendes zu achten:

- Fachkräfte befragen, die die Tätigkeit auch wirklich ausführen,
- befragte Person reden lassen, nicht unterbrechen, sondern zuhören,
- mittels der Leitfragen unauffällig das Gespräch steuern,
- nicht immer auf Vollständigkeit fixiert sein.

Aus dem *Kontext-Szenario* ***(UKS)*** werden auch *Prüfkriterien* ***(PK.1)*** abgeleitet, die als Input in den UE-Dialektischen Prozess **(UE-D)** gehen, um dort im Dialog mit dem IT-Team erwei-

tert oder ergänzt zu werden und später in die Aktivitäten der Qualitäts-Sicherung **(QS)** einzufliessen.

Ausserdem wird hier mit der Generierung eines *Projekt-Planes **(PPL)*** begonnen, der als Grundlage zur Erstellung des Planungs-Vertrages dient (siehe Anfang dieses Kapitels), und in die Aktivitäten des Planungs-Managements **(PM)** eingeht. Unter anderem soll hier die Projekt-Start-Beschreibung soweit ausgereift sein, dass man die Projekt-Mannschaft (mit Disziplinen, Abteilungen u.ä.), die Zuständigkeiten, Verantwortungsträger, verschiedene Ressourcen und alle Beteiligten an dem Software-Entwicklungs-Prozess benennen kann (siehe Anfang dieses Kapitels).

Bei der Ableitung der Kontext-Anforderungen an das Produkt in der Aktivität Kontext-Anforderungen **(KS-A)** ableiten, wird das Kontext-Szenario mittels eines Auswertungsschemas ausgewertet (siehe Abbildung 4-8, ausführlicher siehe Kapitel 2).

Kontext-Szenario	**Erfordernisse aus der Aufgabe**	**Dialogprinzip**	**Anforderungen an den Dialog**	**Prüfkriterium**
...	...	...	...	...

Abbildung 4-8 Kontext-Szenario: Auswertungsschema

Die Aufgaben werden in ihrer Wechselwirkung mit anderen Merkmalen des Nutzungs-Kontextes beschrieben. Aus diesen analytischen Daten werden konstruktive Anforderungen an den Dialog des Benutzers mit dem SWP hergeleitet, welche in der Aktivität **(UK)** erprobt werden.

PRODUKTFLUSS **(UA)**:

von Aktivität	*Produkt*	**nach Aktivität**
-	*Externer Input(EI)*	KS-E
KS-E	*Kontext-Szenario (UKS)*	KS-A
KS-A	*Nutzungs-Beschreibung(UNB)*	UE-D
KS-A	*Nutzungs-Beschreibung(UNB)*	SM-A
KS-A	*Prüfkriterien(PK.1)*	UE-D
KS-A	*Nutzungs-Kontext(UNK)*	UK
PPL	*Projektplan(PPL)*	PM

Die Konstruktive Usability (UK)

Die Konstruktive Usability besteht aus der Aktivität Use-Case **(UC)**. Der Input sind hier die Kern-Aufgaben (aus dem o. g. Auswertungs-Schema) des *Nutzungs-Kontextes **(UNK)***, die in Use-Cases unter der Einflussnahme der *Validierten Anforderungen **(UE-A)*** aus **UE-D** konkretisiert werden. Dieser Input gilt für **(UK)** als eine Menge der **(UC),** weil für jede Zielgruppe und jede Kern-Aufgabe jeweils ein **(UC)** erstellt wird. Sie beinhalten den Entwurf der

Ausführung der Kern-Aufgaben der Benutzer in Form von Dialog-Aktionen. Ein Vorgehensbeispiel wurde in Kapitel 2.2.3. (WinZip) und in Kapitel 5 (ZENO) vorgestellt.

Die Aufgabe der Konstruktiven Usability ist es, die U-Analyse des *Nutzungs-Kontextes **(UNK)*** in Entwurfsvorschläge umzusetzen. Eine erprobte Entwurfsgrundlage dafür sind *Use-Szenarien **(US)*** (mittels Use-Cases). Use-Szenarien dienen der Erfassung der Interaktion des Benutzers mit der Software unter den gegebenen Nutzungsbedingungen am Bildschirmarbeitsplatz (falls „alte SW“ vorhanden). D.h. Voraussetzung zur Entwicklung der Use-Szenarien ***(US)*** ist die Erhebung des *Nutzungs-Kontextes **(UNK)*** mittels *Kontext-Szenario **(UKS)*** in der Analytischen Usability **(UA)**.

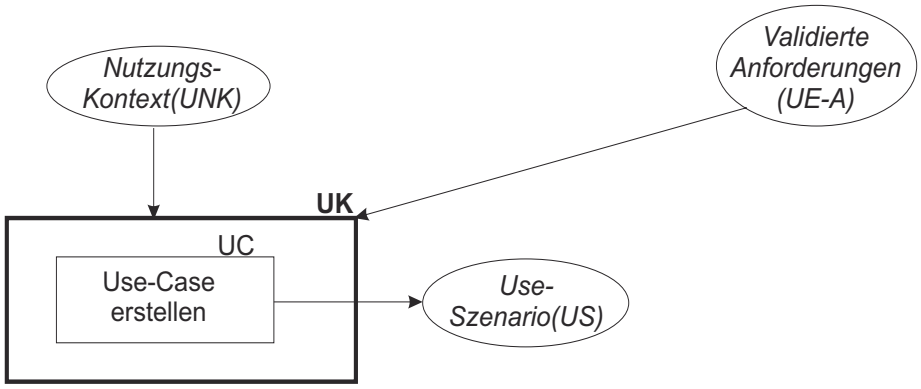

Abbildung 4-9 Konstruktive Usability (UK)

In der **(UK)** soll die Aufgabe in Teilaufgaben zerlegt werden, damit man den Ablauf der Arbeit dokumentieren und besser nachvollziehen kann. Das heisst, die Interaktion des Benutzers wird für jede Kern-Aufgabe ("**key task**") vollständig beschrieben, damit die aus dem Szenario abgeleiteten Anforderungen an das Dialogsystem mit den am Prototypen realisierten Merkmalen verglichen werden können (siehe Abbildung 4-10).

Das *Use-Szenario **(US)*** geht als Dokumentation eines Usability-Prototyps in Form eines **"Lasten-Heft"** (Tätigkeits-Anforderung) als Input in die Aktivität UE-Synthese-Prozess **(UE-S)** ein, dort tritt es dem **„Pflichten-Heft“** (Merkmals-Anforderung) des System-M-Entwurfs **(SM-E)** gegenüber. Die Auswertung der *Use-Szenarien **(US)*** fliesst dort in das Explorative Prototyping **(UE-EP)** ein.

Aufgabe in Teilaufgaben zerlegen	***Use-Case***		**Anforderungen oder zu klärende Fragen**
	Benutzer-Aktion	**Aktion/ Reaktion des Dialog-systems**	
…	…	…	…

Abbildung 4-10 Use-Szenario: Auswertungsschema

Zu Beginn der Erhebung des Use-Szenarios wird dem zu beobachtenden Benutzer erläutert, dass es sich um eine Beobachtung handelt, in der der Beobachter versucht, den Arbeitsablauf am Prototyp zu verstehen und zu dokumentieren. Es soll klar gestellt werden, dass es nicht um eine Bewertung der Arbeit geht, erst recht nicht um die Bewertung des Benutzers als Person, sondern vielmehr um die alleinige Bewertung des Softwaredialoges. Der Benutzer soll ermuntert werden, während der Beobachtung persönliche Einschätzungen und Verbesserungsmöglichkeiten zu äussern. Nach dem Protokollieren sollen die aufgezeichneten Sachverhalte von dem Benutzer auf Richtigkeit geprüft werden und alle Ergebnisse zur Validierung vorgelegt werden, damit Missverständnisse beseitigt werden können.

PRODUKTFLUSS (**UK**):

von Aktivität	*Produkt*	**nach Aktivität**
KS-A	*Nutzungs-Kontext(UNK)*	UK
UK	*Nutzungs-Kontext(UNK)*	UC
UE-D	*Validierte Anforderungen(UE-A)*	UK
UC	*Use-Szenario(US)*	UE-S

4.2.2 Die Anforderungs-Entwicklung Aktivität

Die (Haupt-)Aktivität Anforderungs-Entwicklung **(UE-AE)** besteht aus drei Aktivitäten: Dem UE-Dialektischen Prozess **(UE-D)**, UE-Synthese-Prozess **(UE-S)** und dem UE-Review-Prozess **(UE-R)**. Die Inputs hier kommen aus der Usability-Analyse **(U)** einerseits, und sind die *Nutzungs-Beschreibung **(UNB)***, die *Prüfkriterien **(PK.1)*** und das *Use-Szenario **(US)***; andererseits kommen sie aus der System-Machbarkeitsanalyse **(SM)** und dies sind die *System Idee **(SI)***, der *Lösungs-Vorschlag **(LV)***, die *Fehler-Spezifikation **(FS)***. Die Ergebnisse der Anforderungs-Entwicklung **(UE-AE)** sind *validierte UE-Anforderungen **(UE-A)***, *Prüfkriterien **(PK.2)*** und ***(PK.3)***, *Merkmal-Problem **(UE-MP)*** und als Endergebnis die *Anforderungs-Definition **(UE-AD)*** (siehe Abbildung 4-11).

Zur Vervollständigung der Abbildung sei bemerkt, dass ***(UNB)*** auch in die Aktivität **(SM-A)** einfliesst, in der sie für **SE1.1** bis **SE1.6** (des V-Modells) die dort sog. „Externe Vorgaben (AG)“ ersetzt und das gesamte **SE1** entscheidend beeinflusst. Ausserdem beeinflussen die *validierten UE-Anforderungen **(UE-A)*** die Konstruktive Usability **(UK)** und System-M-Entwurf **(SM-E),** wo sie eine grosse Rolle spielen.

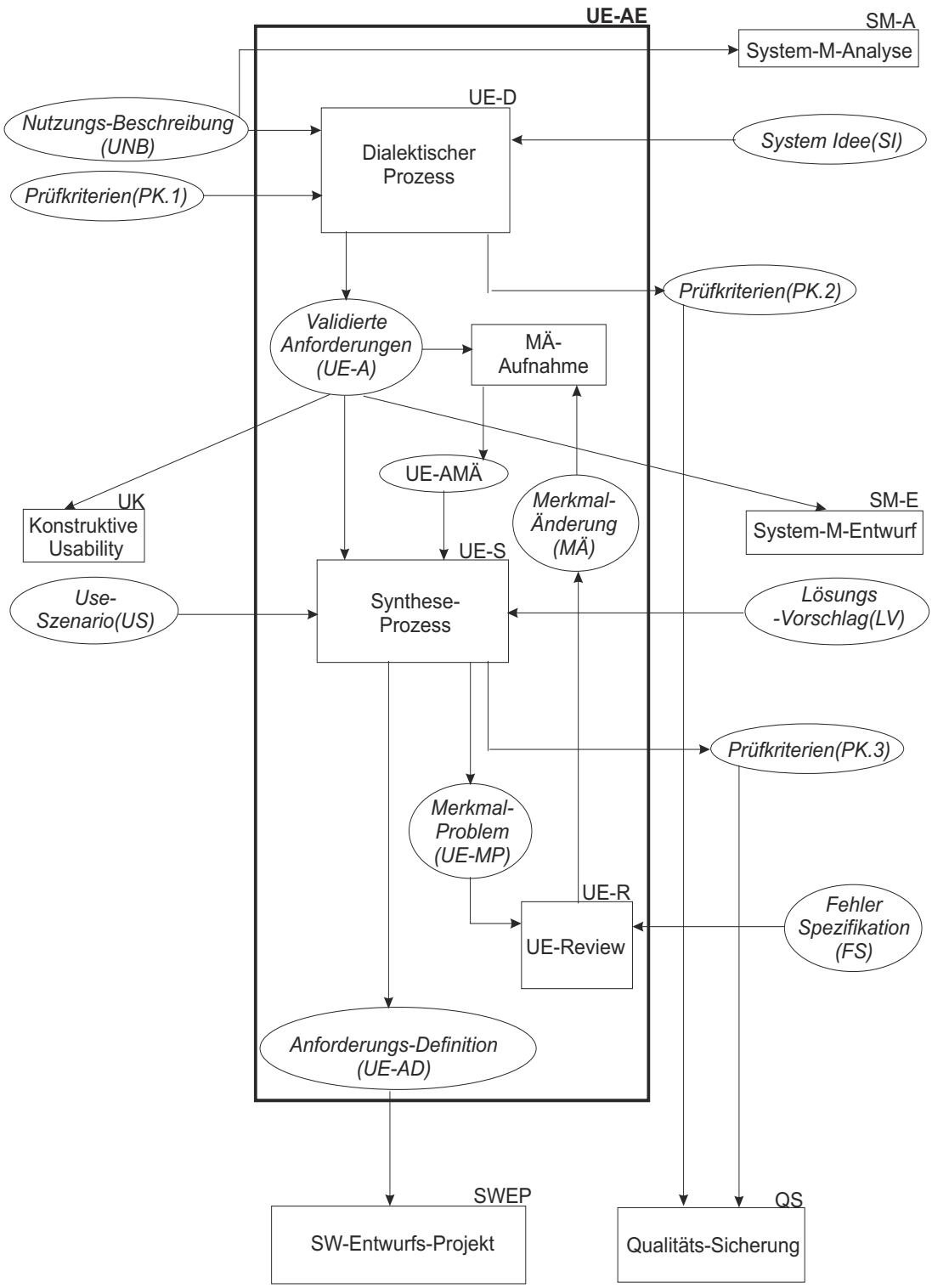

Abbildung 4-11UE-Anforderungs-Entwicklung (UE-AE)

Die Entwicklung der *Anforderungs-Definition* ***(UE-AD)*** wurde im Kapitel 2 vorgestellt und diskutiert. Das Ergebnis ***(UE-AD)*** gilt als ultimativer Input für das SW-Entwurfs-Projekt **(SWEP)**(<Entwurfs-Werkstatt>), indem man unter den Vorgaben der *Anforderungs-Definition* erst zum SW-Entwurf übergeht, anschliessend zur SW-Implementierung und zum SW-/UE-Pflegeprozess **(UE-Pfl)**. Von dem letzteren kann man wieder neue Anforderungen in die Usability-Analyse **(U)** einfliessen lassen (für das neue Release). Die *Prüfkriterien* ***(PK.2)*** und ***(PK.3)*** werden in **(QS)** gesammelt, um die Reproduzierbarkeit, die Qualität, Konformität zu den ISO Normen des SWP feststellen zu können.

Das UE-Review (UE-R)

Das UE-Review dient der Merkmals-Behandlung: Die Inputs der **(UE-R)** Aktivität sind *Merkmal-Problem* ***(UE-MP)*** des Synthese-Ergebnisses aus dem Synthese-Prozess **(UE-S)** und die *Fehler-Spezifikation* ***(FS)*** aus Fehlerbehandlung **(SM-F)**. Die Aufgabe der **UE-Review** Aktivität besteht darin, eine „Pro/Contra-Aufstellung" des *Merkmal-Problems* ***(UE-MP)*** des Synthese-Ergebnisses abzuwägen. Zu Grunde dieser Aktivität liegt die Idee der „Claims Analysis" [Carr/Ross´92], die sich nach einer allgemeinen Formel richtet: Eine Technologie hat zugleich wünschenswerte und nicht wünschenswerte Konsequenzen. Das heisst, um zu entscheiden, welche überwiegen, muss man sich in eine „Pro- und Contra-Aufstellung" begeben, die von einem bestimmten Kontext abhängig ist (siehe auch Kapitel 2). Das UE-Review **(UE-R)** ist also eine Massnahme, die man ergreifen muss, um im Kontext Entscheidungen darüber zu treffen, welche Merkmale des zukünftigen SWP einen Nutzen oder keinen Nutzen versprechen, oder noch nicht verstanden worden sind. Das Ergebnis hier ist dann ein *geändertes Merkmal* ***(MÄ)***, das mit den *validierten UE-Anforderungen* ***(UE-A)*** in die MÄ-Aufnahme **(MA-Ä)** eingetragen wird. Diese geänderte Anforderungsaufstellung heisst *Aufgenommene MÄ* ***(UE-AMÄ)***, welche noch an Prototypen in der Aktivität **(UE-EP)** erprobt werden muss, falls der Benutzer den geänderten Lösungsvorschlag nicht versteht oder nicht akzeptiert.

PRODUKTFLUSS (**UE-EA):**

von Aktivität	*Produkt*	**nach Aktivität**
KS-A	*Nutzungs-Beschreibung(UNB)*	UE-D
KS-A	*Nutzungs-Beschreibung(UNB)*	SM-A
KS-A	*Prüfkriterien(PK.1)*	UE-D
UK	*Use-Szenario(US)*	UE-S
SM-A	*System Idee(SI)*	UE-D
SM-E	*Lösungs-Vorschlag(LV)*	UE-S
SM-F	*Fehler-Spezifikation(FS)*	UE-R
UE-D	*Validierte Anforderungen(UE-A)*	UE-S
UE-D	*Validierte Anforderungen(UE-A)*	UK
UE-D	*Validierte Anforderungen(UE-A)*	SM-E
UE-D	*Prüfkriterien(PK.2)*	QS
UE-S	*Anforderungs-Definition(UE-AD)*	SWEP
UE-S	*Merkmal Problem(MP)*	UE-R
UE-R	*Merkmal-Änderung(MÄ)*	MÄ-A
UE-D	*Validierte Anforderungen(UE-A)*	MÄ-A
MÄ-A	*Aufgenommene MÄ(UE-AMÄ)*	UE-S
UE-S	*Prüfkriterien(PK.3)*	QS
UE-R	*Merkmal Änderung(MÄ)*	UE-A

Der UE-Dialektische Prozess (UE-D)

Die Prinzipien des Dialektischen Prozesses wurden schon im Kapitel 4.1 ausführlich vorgestellt. Die Frage, die man hier stellt, lautet: "*Was soll man erreichen*?"

Das Ziel dieses Prozesses ist das Verstehen der *Nutzungs-Beschreibung* ***(UNB)*** durch das IT-Team und der Beschreibung der *System Idee* ***(SI)*** durch das UE-Team. Die Aufgabe des Prozesses ist, die in diesem „Verstehensprozess" aufgetauchten Widersprüche und Missverständnisse, d. h. die „Dialektische Barriere" abzubauen, die Ziele zu präzisieren und die Mittel mit Realisierungsvorschlägen abzustimmen (zu klären, konkretisieren), um so einer konstruktiven Lösung näher zu kommen (übrigens, man darf keinen der Widersprüche übergehen, nur weil einem kein Lösungsvorschlag einfällt).

Der UE-Dialektische Prozess **(UE-D)** beinhaltet zwei Aktivitäten, nämlich den Dialektischen Prozess selber **(UE-D.1)** zur Bewältigung der „Dialektischen Barriere" und die Prüfkriterien-Entwicklung **(PK-2*)***. Inputs sind die *Nutzungs-Beschreibung* ***(UNB)***, die *Prüfkriterien* ***(PK.1)***, die u. a. in dem Kontext-Szenario beschrieben sind (dazu siehe Abbildung 4-12) und *System-Idee* ***(SI)***.

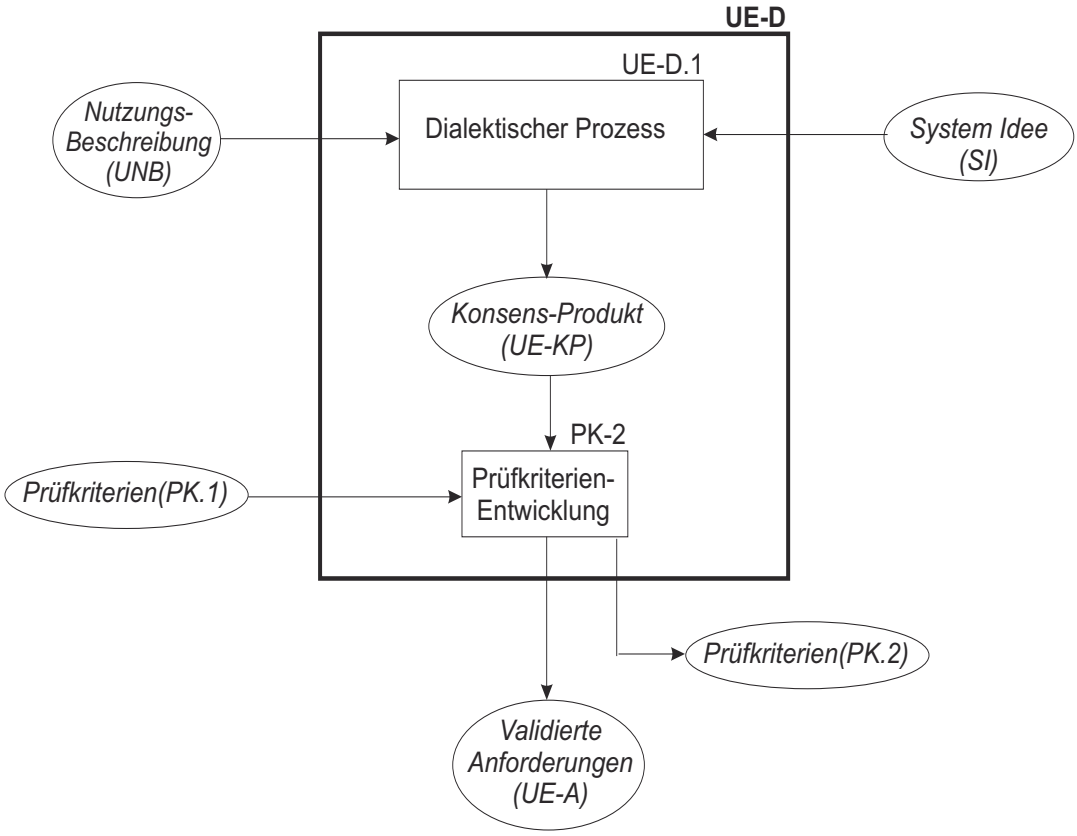

Abbildung 4-12. Der UE-Dialektische Prozess (UE-D)

Das Zwischenergebnis ist ein Produkt der *Konsensfindung **(UE-KP)***, das in die Aktivität Prüfkriterien-Entwicklung **(PK-2)** einfliesst. Dieses entscheidende Zwischenergebnis beinhaltet Informationen über die Tätigkeits-Anforderungen mit dazugehörigen Merkmals-Anforderungen, die sich ursprünglich aus der *Nutzungs-Beschreibung **(UNB)*** und der *System-Idee **(SI)*** der beiden Teams ergaben und unter dem Dialektischen Prozess **(UE-D.1)** einen Wechsel vom **Ist-Zustand** der verstandenen Nutzungsanforderungen seitens des IT-Teams und der verstandenen sw-technischen Möglichkeiten seitens UE-Engineer in den **Soll-Zustand** vollziehen (somit konkretisiert werden). An dieser Stelle wird ein Lösungs-Vorschlag angedeutet, der später in der Aktivität Synthese-Prozess **(UE-S)** erprobt wird.

In der Aktivität Prüfkriterien-Entwicklung **(PK-2)** werden anhand des Zwischenergebnisses ***(UE-KP)*** und der *Prüfkriterien **(PK.1)*** neue *Prüfkriterien **(PK.2)*** entwickelt, gegen die dann das Zwischenergebnis ***(UE-KP)*** geprüft wird.

Das Ergebnis des Prozesses ist eine Aufstellung der *validierten Anforderungen **(UE-A)***, auf die sich die beiden Teams geeinigt haben. Die *validierten Anforderungen **(UE-A)*** sind in Form eines **"Lasten-Heftes"** (Tätigkeits-Anforderung) und **„Pflichten-Heftes“** (Merkmals-Anforderung) dokumentiert (Kapitel 2), sie gehen als Input in die Aktivität UE-Synthese-Prozess **(UE-S)** ein.

PRODUKTFLUSS **(UE-D)**:

von Aktivität	*Produkt*	**nach Aktivität**
UA	*Nutzungs-Beschreibung(UNB)*	UE-D.1
SM-A	*System-Idee(SI)*	UE-D.1
UE-D.1	*Konsens-Produkt(UE-KP)*	PK-2
UA	*Prüfkriterien(PK.1)*	PK-2
PK-2	*Prüfkriterien(PK.2)*	QS
PK-2	*Validierte Anforderungen(UE-A)*	UE-S

Der UE-Synthese-Prozess (UE-S)

Der UE-Synthese-Prozess **(UE-S)** beinhaltet zwei Aktivitäten: Das Explorative Prototyping **(UE-EP)** und die Merkmal-Evaluierung **(UE-ME)**. Der Input sind das *Use-Szenario **(US)*** als ein prototypischer Aufgaben-Ablauf aus der Benutzer-Sicht, die *validierten Anforderungen **(UE-A)*** und der *Lösungs-Vorschlag **(LV)*** (siehe Abbildung 4-13).

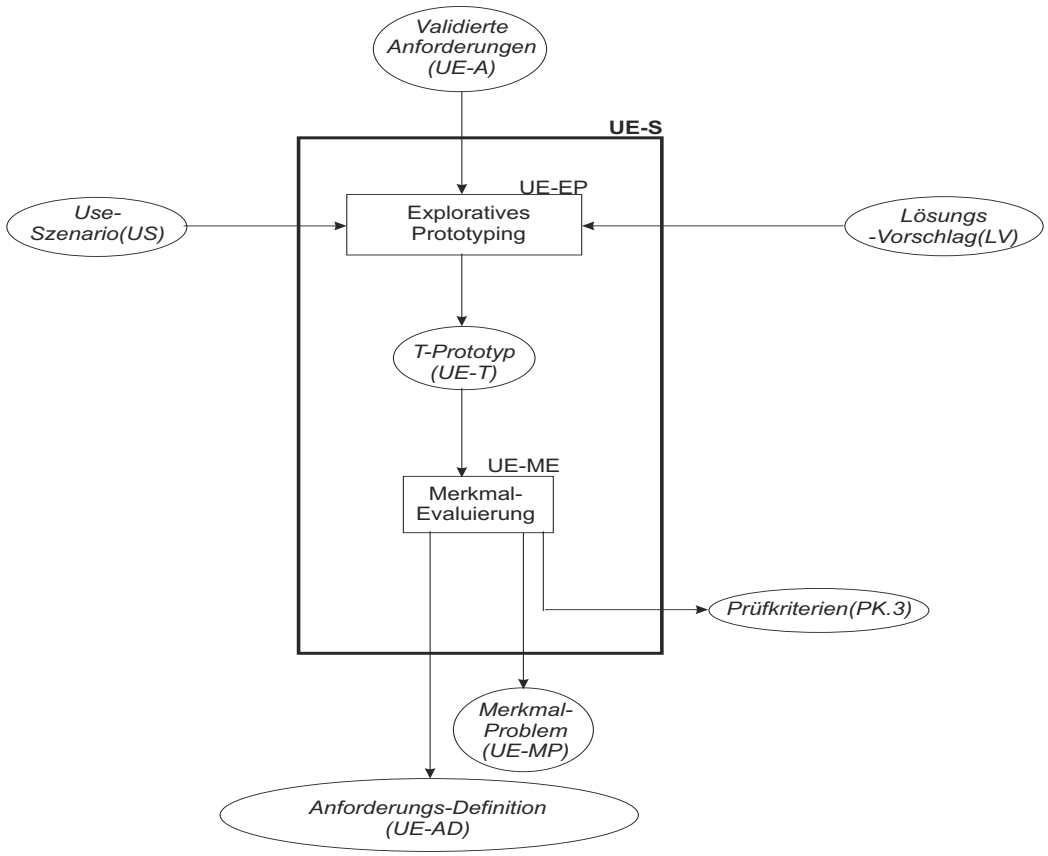

Abbildung 4-13 UE-Synthese-Prozess [UE-S]

Die Frage, die man hier stellt, lautet: "*Womit soll etwas erreicht werden*?"

Zunächst wird ein Exploratives Prototyping **(UE-EP)** (genauer im Kapitel 2) durchgeführt, in dem anhand der Use-Szenarien und der Prüfkriterien SWP-Merkmale erprobt (ausprobiert) werden. Das Ergebnis ist meist ein *T-Prototyp **(UE-T)***, der die Beschreibung des betrachteten Merkmals liefert. Die Merkmal-Evaluierung **(UE-ME)** kann mittels Heuristiken von Nielsen (siehe Kapitel 2) und unabhängigen Prüfverfahren und Bewertungskonzepten, wie **DATech-Prüfhandbuch Gebrauchstauglichkeit** (Leitfaden für die software-ergonomische Evaluierung auf der Grundlage von DIN EN ISO 924/10-11) durchgeführt werden.

Die im DATech-Prüfhandbuch Gebrauchstauglichkeit [DAT-G´00] dargestellte Methodik ist keine Verfahrensanweisung, sondern ein Leitfaden, der auch als Grundlage für die Begutachtung von Prüflaboratorien dient. Prüflaboratorien sollen diesen Leitfaden als Referenzmodell für die Entwicklung ihrer eigenen Prüfmethodik anwenden. „In der Praxis der Prüflaboratorien hat sich der in [ISO-9241] nahe gelegte Verifikationsansatz als zu aufwendig herausgestellt. Softwareprüfer standen stets vor der nicht entscheidbaren Frage, wo eine Prüfung anfangen und aufhören soll. Bei einer entwicklungsbegleitenden Prüfung hingegen ist der Verifikationsansatz weiterhin empfehlenswert, weil bei jeder einzelnen Entwurfsentscheidung die

Konformitätsfrage zu stellen ist. Darüber hinaus kann der Entwickler daran interessiert sein, eine Lösung zu finden, die eine von den Normen empfohlene Mindestanforderung übertrifft.“ Bei festgestellter Non-Konformität (siehe Glossar) einzelner Merkmale soll die erforderliche Verbesserung des Produkts vorbereitet werden.

Das Ergebnis des Prozesses **(UE-S)** ist die *Anforderungs-Definition **(UE-AD)*** (genauer Kapitel 2), *Prüfkriterien **(PK.3)*** und u. U. auch ein *Merkmal-Problem **(UE-MP)***, welches schon zum Anfang dieses Kapitels unter UE-Review **(UE-R)** ausführlich beschrieben wurde.

Es wird eine Dokumentation erstellt, die aus einem **"Lasten-Heft"** (Tätigkeits-Merkmale) und einem **"Pflichten-Heft"** (Produkt-Merkmale) besteht und damit das Ergebnis der Aktivität UE-Synthese-Prozess **(UE-S)** darstellt, das in den SW-Entwurfs-Projekt als Input einfliesst.

An dieser Stelle sei noch bemerkt, dass es einen Sonderfall geben kann, in dem die *validierten Anforderungen **(UE-A)*** direkt als *Anforderungs-Definition **(UE-AD)*** gelten: Wenn die beiden Prozesse **(UE-D)** und **(UE-S)** parallel verlaufen.

PRODUKTFLUSS **(UE-S)**:

von Aktivität	Produkt	nach Aktivität
UK	*Use-Szenario(US)*	UE-EP
UE-D	*Validierte Anforderungen(UE-A)*	UE-EP
SM-E	*Lösungs-Vorschlag (LV)*	UE-EP
UE-EP	*T-Prototyp (UE-T)*	UE-ME
UE-ME	*Prüfkriterien (PK.3)*	QS
UE-ME	*Merkmal-Problem (UE-MP*	UE-R
UE-ME	*Anforderungs-Definition(UE-AD)*	SWEP

4.2.3 Die System-Machbarkeitsanalyse Aktivität

Wie schon in früheren Kapiteln angesprochen, wird anhand des Vorgehensmodells, genauer des Regelungsteils 1 der "Regelungen Submodell Systemerstellung" [Drös´00], ein Vorschlag zur Ergänzung/Modifizierung des V-Modells um Elemente des Usability-Engineering durchgeführt.

Dieser Integrationsversuch der Aspekte des UE mit dem V-Modell beginnt in der „Software-Engineering-Sicht“ (in diesem UE-Prozess-Modell in der System-Machbarkeitsanalyse **(SM)**). An dieser Stelle wird die Abgleichung des V-Modells mit dem UE vorgenommen. Dies führt zur Modifizierung der Submodelle **SE1**, **SE2**, **SE3**, **SE4**, **SE5** um UE-Aspekte, die dann in die Anforderungs-Entwicklungs-Sicht einfliessen, um dort an der gemeinsamen Anforderungs-Definitions-Entwicklung beteiligt zu sein.

Dabei gilt hier die Strategie: Bei der Gestaltung von Softwareentwicklungs-Prozessen muss von Anfang an das Spannungsfeld zwischen Nutzungskonzept (Entwurf intendierter Nutzung anhand von Use-Szenarien) der Usability-Analyse **(U)** und Systemkonzept (Entwurf soft-

ware-technischer Aspekte bis zu Programmiervorgaben), der System-Machbarkeitsanalyse **(SM)** in Betracht gezogen und untersucht werden. Im folgenden ein Überblick über die Aktivität **(SM)** (siehe Abbildung 4-14).

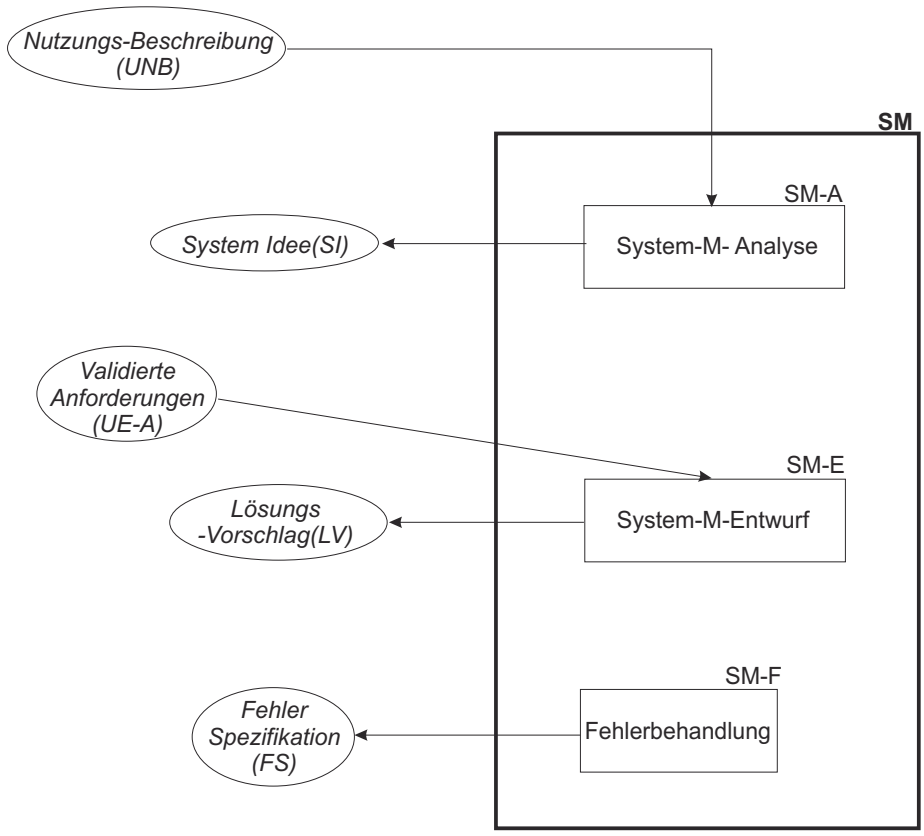

Abbildung 4-14 System-Machbarkeitsanalyse (SM)

Die (Haupt-)Aktivität System-Machbarkeitsanalyse **(SM)** besteht aus drei Aktivitäten: Der System-M-Analyse **(SM-A)**, dem System-M-Entwurf **(SM-E)** und der Fehler-Behandlung **(SM-F)**. Der Input ist zum einen die *Nutzungs-Beschreibung **(UNB)*** und zum anderen die *Validierten Anforderungen **(UE-A)*.** Der Output hier, der dann in die Aktivität Anforderungs-Entwicklung **(UE-AE)** einfliesst, sind die *System-Idee **(SI)***, *Lösungs-Vorschlag **(LV)*** und u. U. die *Fehler-Spezifikation **(FS)***.

Die Fehlerbehandlung (SM-F)

In der Aktivität Fehlerbehandlung **(SM-F)** in dem Abschnitt **SE5.1-SW**: SW-Kompo-nente/ -Modul/ DB des VM muss spezifiziert werden, wenn *ein Merkmal-Problem **(UE-MP)*** aus der Aktivität Merkmal-Evaluierung **(UE-ME)** auftaucht. Diese *Fehler-Spezifikation **(FS)*** fliesst

als Input in die **UE-Review** Aktivität ein, um dort bestimmte Merkmale abzustimmen/ggf. zu ändern (dazu siehe Abbildung 4-14).

PRODUKTFLUSS **(SM)**:

von Aktivität	*Produkt*	**nach Aktivität**
UA	*Nutzungs-Beschreibung (UNB)*	SM-A
UE-D	*Validierte Anforderungen(UE-A)*	SM-E
SM-A	*System-Idee (SI)*	UE-D
SM-E	*Lösungs-Vorschlag (LV)*	UE-S
UE-ME	*ggf. Merkmal-Problem (UE-MP)*	SM-F
SM-F	*ggf. Fehler-Spezifikation (FS)*	UE-R

Die System-Machbarkeits-Analyse (SM-A)

Die System-M-Analyse **(SM-A)** Aktivität besteht aus System-Anforderungsanalyse **(SE1)** und SW/HW-Anforderungsanalyse **(SE3)**. Das Ergebnis ist die IT-System-Beschreibung in Form einer *System-Idee **(SI)***. Dieses Ergebnis fliesst in die Aktivität UE-Dialektischer Prozess **(UE-D)** ein, um dort der *Nutzungs-Beschreibung **(UNB)*** der U-Kontext-Analyse **(UA)** gegenüber gestellt zu werden (siehe Abbildung 4-15).

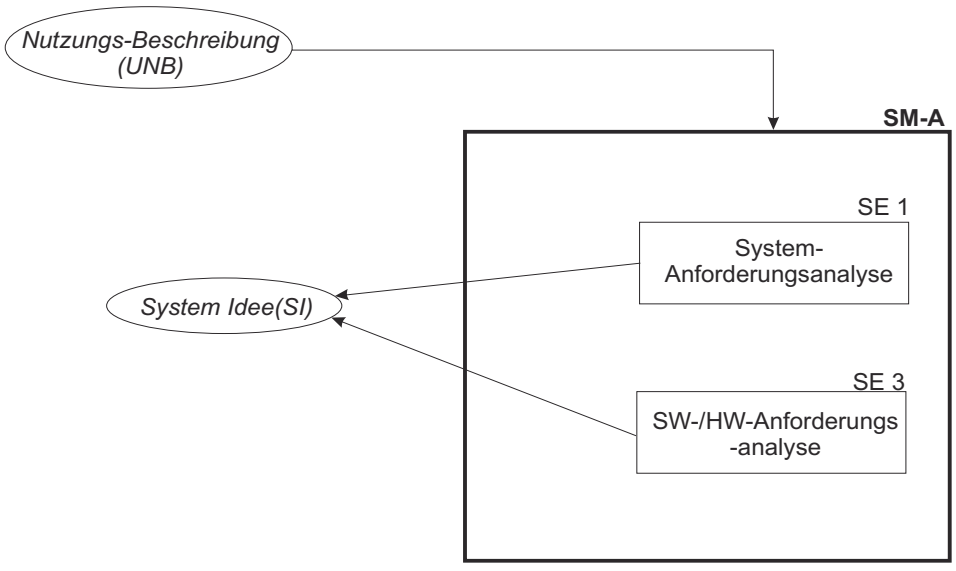

Abbildung 4-15 System-Machbarkeits-Analyse (SM-A)

In **SE1** gibt es zwei modifizierte Stellen: *Externe Vorgaben (AG)* werden ersetzt durch *Nutzungs-Beschreibung **(UNB)*** und die *Anwenderforderungen* durch (zunächst) *System Idee erster Stufe **(SI-1)***. Die ***(SI-1)*** (bzw. auch ***(SI)***) ist in einer Tabellenform, in der die entsprechenden Produkt-Merkmale, die sich aus den Teilaktivitäten der **SE1** ergaben, den Tätigkeits-Anforderungen aus ***(UNB***) gegenüber gestellt werden. Falls die Gegenüberstellung ausrei-

chend für die **UE-D.1** ist, und man keine SW/-HW-Analyse des **SE3** mehr für SI-Entscheidung benötigt, dann gilt die ***(SI-1)*** auch als ***(SI)***. Wenn der andere Fall eintritt, dann fliesst ***(SI-1)*** in **SE3** und ersetzt dort die *Anwenderforderungen*. Dann gibt es in SE3.5 einen zusätzlichen Output, der *System Idee* ***(SI)*** heisst.

Die Aufgabe hier ist, eine System-Idee vorzuschlagen, die sich aus der Benutzertätigkeit herauskristallisiert und nicht nur aus sw-technischen Anforderungen ergibt.

PRODUKTFLUSS **(SM-A)**:

von Aktivität	*Produkt*	**nach Aktivität**
UA	*Nutzungs-Beschreibung(UNB)*	SE1
SE1	*System Idee (SI)*	UE-D.1
SE1	*ggf. System Idee erster Stufe(SI-1)*	SE3
SE3	*ggf. System Idee (SI)*	UE-D.1

Der System-Machbarkeits-Entwurf (SM-E)

Die System-Entwurf Aktivität **(SM-E)** beinhaltet zwei Aktivitäten: **SE2**, **SE4**. Anhand der *Validierten Anforderungen* ***(UE-A)*** wird ein *Lösungs-Vorschlag* ***(LV)*** in Form eines Prototyps aus der Sicht des IT-Entwicklers erstellt, der in dem explorativen Prototyping **(UE-EP)** gegenüber den *Use-Szenarien* ***(US)*** erprobt wird (dazu siehe Abbildung 4-16).

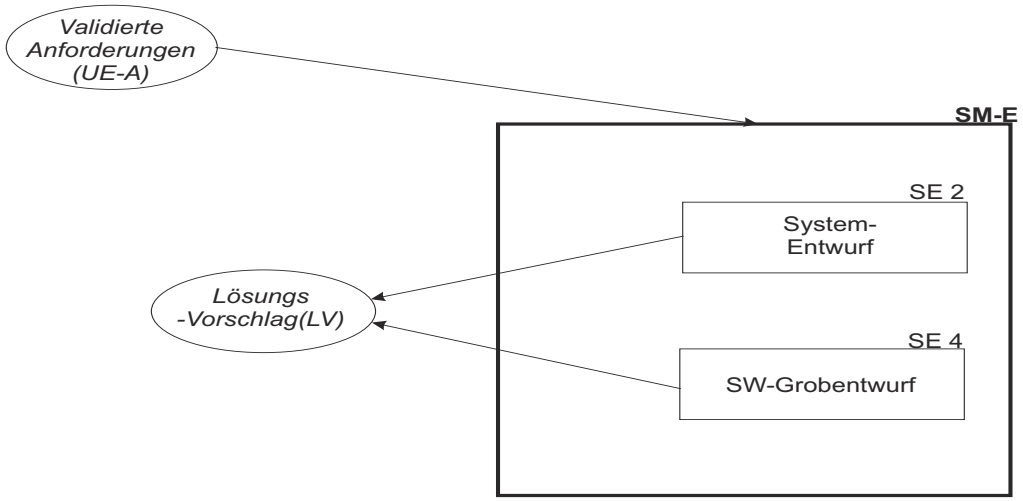

Abbildung 4-16 System-Machbarkeits -Entwurf (SM-E)

In **SE2** und **SE4** gibt es jeweils eine modifizierte Stelle, nämlich die *Anwenderforderungen* werden durch die sich aus dem **UE-D** ergebenden *Validierten Anforderungen* ***(UE-A)*** ersetzt. Je nach Verfeinerungsbedarf resultiert der *Lösungs-Vorschlag* ***(LV)*** aus der **SE2** oder **SE4.**

Die Aufgabe ist hier also einen Lösungsvorschlag (Systementwurf) zu machen, der sich aus den *Validierten Anforderungen **(UE-A)*** herauskristallisiert und nicht nur aus sw-technischen Anforderungen ergibt.

PRODUKTFLUSS **(SM-E)**:

von Aktivität	*Produkt*	**nach Aktivität**
UE-D	*Validierte Anforderungen (UE-A)*	SE2
UE-D	*Validierte Anforderungen (UE-A)*	SE4
SE2	*Lösungs-Vorschlag (LV)*	UE-S
SE4	*ggf. Lösungs-Vorschlag (LV)*	UE-S

4.2.4 Die SW-Pflege Aktivität

Diese Aktivität **(UE-Pfl)** beschäftigt sich mit der Weiterentwicklung des SWP aus der Nutzung und ist Teil des *SW-Einführungs-Projektes* (siehe Anfang des Kapitels). Sie unterstützt die Aktivität *Überleitung in die Nutzung* **SE9***,* genauer gesagt, sie liefert Ergänzungen zu **SE9.3**, um einen Beitrag für die Pflege- und Wartung des SWP im Sinne des UE zu leisten (u. a. Nutzungs- und Einarbeitungs-Probleme zu lösen).

Hier unterscheidet man zwischen *Fehlern*, *Mängeln* und *Schwachstellen* des SWP, die entsprechend bewertet, behandelt und verbessert werden, um so das SWP aus der Nutzung her für die neue Version vorbereiten zu können (siehe Abbildung 4-17). Im Verbesserungs-Prozess **(V)** betreffen z. B. *Fehler **(F)*** „bugs“ in der Software, *Mängel **(M)*** betreffen Nutzungs-Probleme ergonomischer Art, und *Schwachstellen **(S)*** betreffen die unzureichende organisatorische Einbettung der Software in die Anwenderorganisation (im Sinne der Anpassung der Organisation an das SWP). All dies wird im Verbesserungs-Prozess **(V)** behoben und anschliessend aufbereitet für die Anforderungen ***(QAE)*** der neuen Version des SWP.

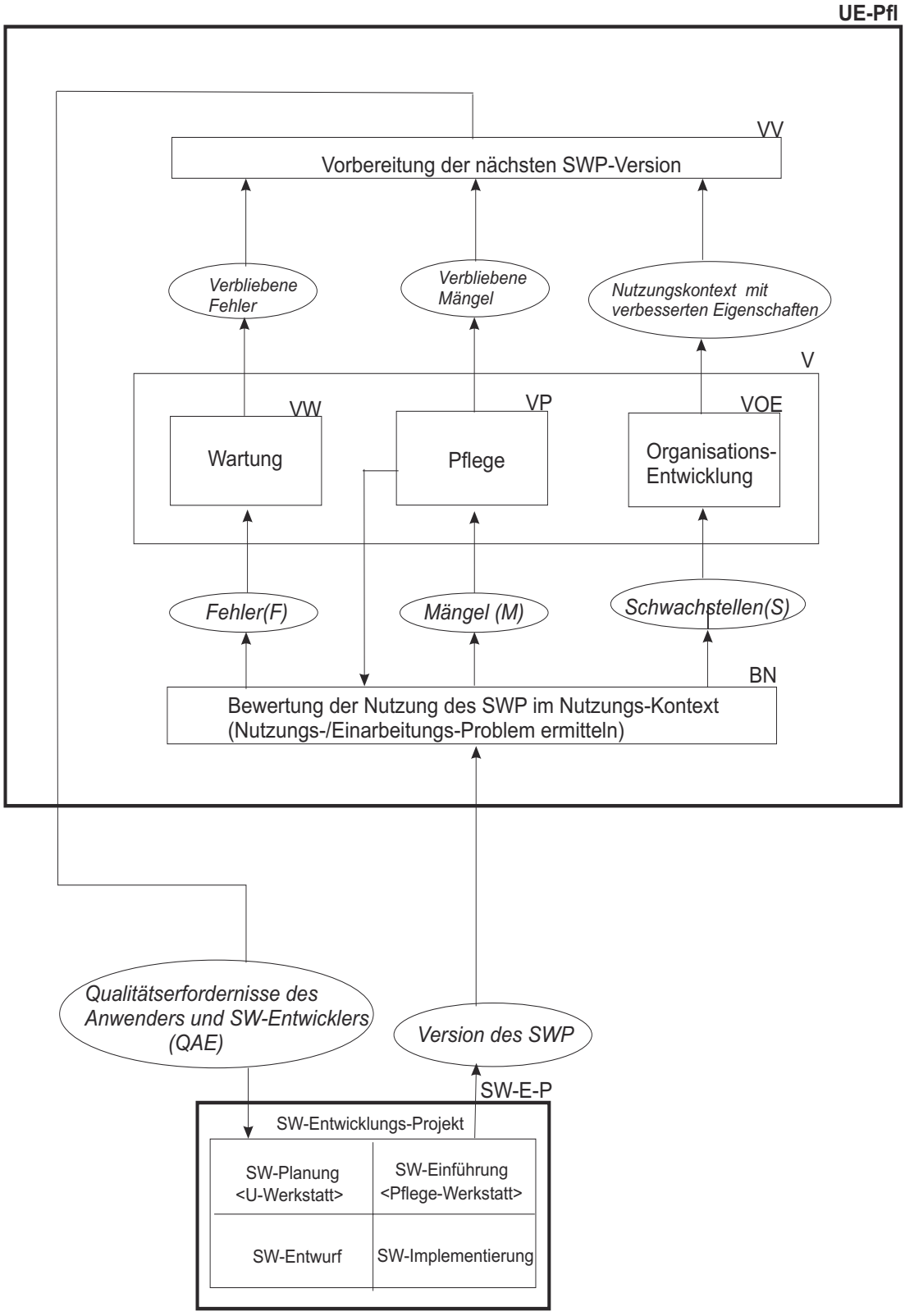

Abbildung 4-17 UE SW-Pflege (UE-Pfl)

Aus UE-Sicht gibt es wichtige Gründe dafür, dass die Anforderungs-Definition für die neue SWP-Version im **(UE-Pfl)** aktualisiert werden muss:

- der Benutzer kann den Prototypen erst jetzt in seiner Arbeitssituation nutzen (verwenden);
- beim Prototyping können meist nur Einarbeitungs-Probleme erkannt werden;
- es können Benutzerrollen hinzukommen, die beim Prototyping nicht beteiligt waren;
- selten vorkommende Aufgaben haben keinen Einfluss auf die Evaluierung,
- Änderung des Nutzungs-Kontextes (neue/andere Aufgaben, Arbeitsabläufe, technische Umgebung)

PRODUKTFLUSS **(UE-Pfl)**:

von Aktivität	*Produkt*	**nach Aktivität**
SE9.3	*Version des SWP*	BN
BN	*Fehler(F)*	VW
BN	*Mängel(M)*	VP
BN	*Schwachstellen(S)*	VOE
VP	*Mängel(M)*	BN
VW	*Verbliebene Fehler*	VV
VP	*Verbliebene Mängel*	VV
VOE	*Verbliebene Schwachstellen*	VV
VV	*Qualitätserfordernisse des Anwenders und SW-entwicklers(QAE)*	U

Ein Überblick über das gesamte UE-Prozess-Modell

Zum Abschluss ein Überblick über das gesamte **UE**-Prozess-Modell in einer Abkürzungsform. Daran wird deutlich, dass die Integration des UE-Engineering und SW-Engineering im Prozess möglich ist. In dieser Diplomarbeit wurde die Ergänzung des V-Modells um wichtige Aspekte des Usability-Engineering durchgeführt (siehe Abbildung 4-18). Um Qualität produzieren zu können, muss das künftige Engineering von SW-Projekten aus UE-Teil und SE-Teil (hier SM) bestehen.

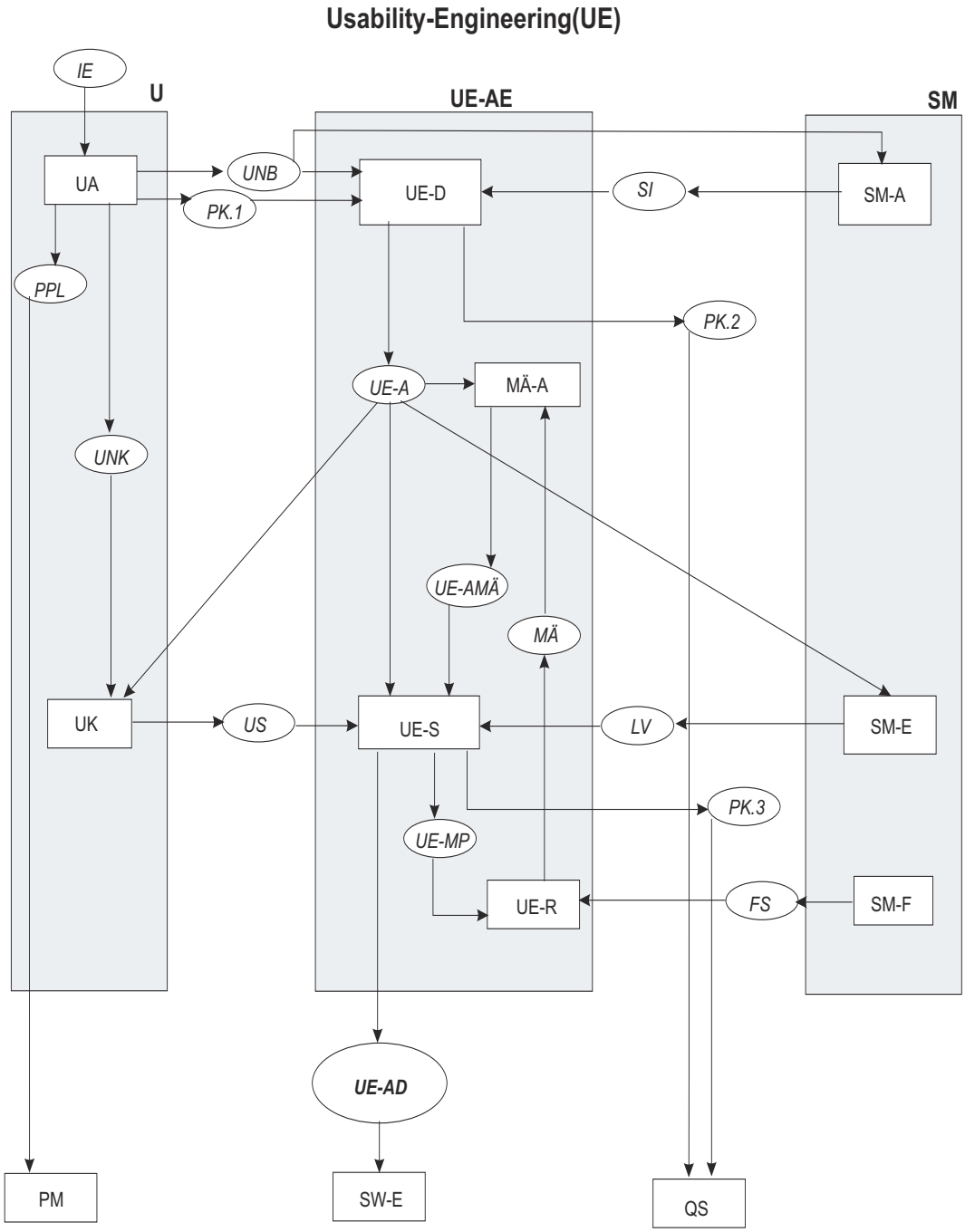

Abbildung 4-18 Überblick über das gesamte UE-Prozess-Modell in Abkürzungsform

Da diese Erweiterung in Form einer zum V-Modell konformen Syntax formuliert ist, kann die Umsetzbarkeit in konkreten Projekten, die gemäss V-Modell durchgeführt werden, evaluiert werden.

Zum Schluss sei nochmals darauf hingewiesen, dass bei der Beurteilung vom SWP sein Vorbereitungs-(Planungs-)Prozess im Vordergrund steht. Für eine SW-Entwicklung müssen im Prozess Voraussetzungen geschaffen werden, um ein gebrauchstaugliches SWP im Sinne ISO 9241-11 produzieren zu können. Das bedeutet, dass die Gestaltung der Gebrauchstauglichkeit von SWP vor allem von ihrem UE-Prozess abhängt. Dabei ist die Reproduzierbarkeit dieses Prozesses von enormer Bedeutung [Chris/Curti'95].

Fazit ist zum einen die Aussage, dass man *aus dem Prozess auf die Qualität des Produktes schliessen kann, aber nicht umgekehrt.* Zum anderen ist das Ziel, einen Rahmen zu schaffen, um die Prozesse ständig verbessern können zu dürfen, d. h. *man begibt sich in einen stetig fortlaufenden Verbesserungs-Prozess hinein:*

„Wir tätigen etwas heute, um es morgen noch besser zu tun“.

Diese Erkenntnisse verlangen von vielen Anwendern und Herstellern, die nur der Qualität eines vorhandenen SWP vertrauen, ein Umdenken. Ausserdem benötigt man zur Durchführung solcher um das UE erweiterten SW-Entwicklungs-Prozesse keine einzelne Genies, sondern fähige Teams, die die Benutzer mit einbeziehen.

4.3 Die Zertifizierung, ISO-Normen und das UE-Prozess-Modell

Mit der SW-Qualitätssicherung und Prüfung wird der Begriff „Zertifizierung“ (Bescheinigung) schon lange diskutiert. Zertifikate aber sind nur dann aussagefähig, wenn die zugrundegelegten Verfahren und Beurteilungen offen gelegt werden, und die Begründungen für ein positives oder negatives Urteil nachvollzogen werden können. Der Begriff „Zertifizierung“ wird für zwei verschiedene Prüf- und Begutachtungsverfahren verwendet, nämlich einerseits für Produkt-Zertifizierung und andererseits für Prozess-Zertifizierung. Bei der Prozess-Zertifizierung wird bescheinigt, dass ein SW-Hersteller Massnahmen ergriffen hat, die mittels von Usability-Engineering erwarten lassen, ein gebrauchstaugliches SWP herzustellen. Die Zertifizierung nach ISO 900x steht für solche Verfahren. Für den Bereich UE-Prozess steht die [ISO-13407] („human-centred design processes for interactive systems“) mit Anforderungen an den Entwicklungs-Prozess und CMM („Capability Maturity Model“) [Thal'00], [ISO-13407], [Chris/Curti'95]. Bei der Produkt-Zertifizierung nach [ISO-9241] (im Wesentlichen von 10 bis17) [ISO-9241-x] zählt allein die erreichte Produktqualität - unabhängig von den Umständen der Herstellung (Herstellungsprozess). Das heisst, es können hier keine Aussagen über die Qualität der zukünftigen Produkte eines Herstellers getroffen werden.

Die Konformität mit sw-ergonomischen Normen wird als Indikator dafür angesehen, dass von einem SWP keine unnötigen Belastungen im Sinne der Bildschirmarbeitsverordnung ausgehen und die Arbeitsaufgaben effektiv und effizient erledigt werden können, was zur Steigerung der Produktivität beiträgt.

In Deutschland befasst sich mit der Einführung eines SW-Qualitäts-Zertifikates die „Deutsche Akkreditierungsstelle für Informations- und Telekommunikationstechnik in der TGA“, dessen Kooperationsmitglieder die DATech-Handbücher für die SW-Begutachtung herausgeben [DATech-UE'00], [DATech-G'00].

Die Prüfgrundlagen zur Beurteilung eines UE-Prozesses sind [ISO-13407] und das CMM, das in den USA der Zertifizierung (Registrierung) von SW-Entwicklungs-Prozessen zu Grunde gelegt wird. In ISO 9000 wird nur empfohlen, „*WAS* in einem QS-Prozess geschehen soll, wobei das *WIE* dem Hersteller überlassen bleibt. Indem man aber im CMM-Modell den Beweis der Effektivität von QS-Massnahmen verlangt, wird mehr über die Dokumentation des WIE, also das Zustandekommen von Ergebnissen, abverlangt, ohne dies ausdrücklich vorschreiben zu müssen. Die Qualitätskriterien eines UE-Prozesses lassen sich in Form von Ergebnissen definieren. Damit weist das CMM-Modell für das UE wesentliche Vorteile auf, die eine Beurteilung der Prozessqualität erleichtern. Eine Herstellerorganisation soll das Modell auch dazu benutzen, den Stand der Reife eines SW-Entwicklungs-Prozesses selbst einzuschätzen (maturity questionnaire), um schrittweise an der Verbesserung des Prozesses zu arbeiten.

Ein weiterer Vorteil des CMM-Modells besteht darin, dass CMM Empfehlungen enthält, wie der Prozess verbessert werden kann. Da Verbesserungen schrittweise (evolutionär) geschehen, weil sie u.U. lange Vorbereitungen brauchen und einige Schritte auf vorangegangenen Schritten aufbauen, wurde das CMM-Modell als ein fünfstufiger Reifungsprozess konzipiert. Für jedes Niveau wird eine begrenzte Anzahl von QS-Massnahmen und Tätigkeiten (key process areas, key practices) definiert, deren Effektivität ganz oder teilweise nachgewiesen werden muß, z.B. anhand von Prozess- oder Produkteigenschaften (common features)„ [DATech-UE'00].

Für die Beschreibung der Reife eines UE-Prozesses reichen drei Stufen aus. Diese Stufen eines "Usability Engineering Maturity Model" **(UEMM)** werden im DATech definiert [DAT-UE'00]:

Stufe 1: Anfangsstadium (initial level),
Stufe 2: Reproduzierbare Effektivität,
Stufe 3: Kontinuierlich sich verbessernder Prozess.

Auf *Stufe 1* ist „Usability-Engineering ein noch nicht von Anfang an berücksichtigter Teil des Software-Entwicklungsprojekts. Das Projekt ist vollkommen damit ausgelastet, überhaupt eine zuverlässige Funktionalität herzustellen. QS-Massnahmen konzentrieren sich auf das Testen. Es fehlen UE-Fachleute im Projekt. Für ergonomische Bewertungen werden allenfalls Berater hinzugezogen.“

Auf *Stufe 2* ist „Usability-Engineering zwar ein wichtiger Teil der Anforderungs-Entwicklung und des Entwurfs. Aber der dem Projekt angehörende Usability-Engineer wirkt hauptsächlich beim Oberflächen-Entwurf mit. ergonomischer Bewertungen sind noch zu wenig kontextbezogen. Man konzentriert sich auf Konformität mit dem Style-Guide. Bei Entwurfs-Entscheidungen hat die Gebrauchstauglichkeit des Produkts geringste Priorität.“

Auf *Stufe 3* ist „der UE-Engineer zusammen mit einem Analytiker von Anfang an fest in das SW-Entwicklungs-Projekt integriert. Der UE-Engineer moderiert die Kontakte zwischen allen an den Entwurfsvorbereitungen beteiligten Personen“: U-Analytiker, U-Prüfer, Benutzer/Anwender und dem IT-Team. Alle erzielten Verbesserungen und Usability-Entscheidungen werden so dokumentiert, dass ein Erfahrungstransfer von Projekt zu Projekt möglich ist und somit die Qualität des UE-Prozesses mehr von der *Projekt-Praxis* als von Personen abhängt. Die Form des Erfahrungstransfers unterstützt letztlich einen sich stetig verbessernden UE-Prozess [DATech-UE'00].

Der in dieser Diplomarbeit vorgestellte UE-Prozess befindet sich auf der **Stufe 3** der Prozess-Reife des „Usability Engineering Maturity Model“ **(UEMM)**.

Konformität des UE-Prozesses zu den ISO Normen

Bei der Erstellung dieses UE-Prozess-Modells wurde stets auf die Einhaltung der Konformität zu den ISO Normen 13407, 12207, 900x geachtet.

Das UE-Prozess-Modell zeigt aber vor allem, wie die in ISO 13407 gestellten Forderungen umgesetzt wurden. Die Forderungen wurden schon in Kapitel 2 beschrieben und die Forderungen aus ISO 12207 in Kapitel 3. Die wesentlichen sind zur Erinnerung in der Abbildung 4-19 aufgestellt. Zur Einhaltung der Konformität mit den Normen ISO 13407,12207 wurden den Aktivitäten des UE-Prozess-Modells die entsprechenden Punkte der Normen zugeordnet (siehe Abbildung 4-20).

ISO 13407	5 bzgl. multidisziplinäre Team-Zusammenstellung, 6 bzgl. Planung 7 bzgl. human-centred design processes for interactive system
ISO 12207	5 bzgl. Primary Processes 6 bzgl. Supporting Processes (v. a. Dokumentation, Prüfungsdokumentation, Verifikation, Validierung) 7 bzgl. Organizational Processes (v. a. Management, Training, Verbesserung, Infrastruktur)

Abbildung 4-19 Relevante Punkte der Normen ISO 13407 und ISO 12207

Aktivität	ISO 13407	ISO 12207
Usability-Engineering (UE)	Pkt: 5, 6, 7.x	Pkt: 5, 6
Analytische Usability (UA)	Pkt: 5, 6, 7.2	Pkt: 5, 7
Konstruktive Usability (UK)	Pkt: 7.2	Pkt: 5
Dialektischer Prozess (UE-D)	Pkt: 7.3	Pkt: 5, 6
Synthese-Prozess (UE-S)	Pkt: 7.3, 7.4	Pkt: 5, 6
Usability-Review (UE-R)	Pkt: 7.5	Pkt: 6, 7
System-Machbarkeits-Analyse (SM-A)	Pkt: 7.2	Pkt: 7
System- Machbarkeits-Entwurf (SM-E)	Pkt: 7.3	Pkt: 5, 7
SW-Pflege (UE-Pfl)	Pkt: 7.5	Pkt: 5, 7

Abbildung 4-20 Tabelle die Einhaltung der Konformität mit den ISO 13407, ISO 12207

Ausserdem lassen sich aber „Unvereinbarkeiten“ mit ISO 9001 sowie ISO 9000-3 in folgenden Punkten feststellen, weil diese Normen nur wenig Berührung zum UE aufweisen:

- Iteration bei der Entwicklung von Anforderungen und beim Entwurf von Software,
- Prototyping,

- Benutzerbeteiligung,
- Anforderungs-Validierung vor der Merkmals-Verifikation,
- Projektmodell als "design-use cycle" und
- nachträgliche Systemanpassung aufgrund der erst im Nutzungs-Kontext eines Produkts feststellbaren Nutzungsprobleme.

Dies zeigt deutlich, dass ganzheitliches Qualitätsmanagement mittels UE notwendig und unabdingbar ist - vergleiche mit Abbildung.1-4 Ganzheitliches Qualitätsmanagement mittels UE aus Kapitel 1.

5 Evaluierung

In diesem Kapitel wird die praktische Umsetzbarkeit des UE-Prozess-Modells und der in ihm verankerten Erweiterung des V-Modells aus dem vorherigen Kapitel demonstriert. Anhand eines praxisbezogenen Kontextes wird das UE-Prozess-Modell evaluiert und die praktische Vorgehensweise dokumentiert.

Das SW-Planungs-Projekt

Wie schon in Kapitel 4 erwähnt, wird in dem *SW-Planungs-Projekt* die Anforderungs-Definition (ein Lösungs-Vorschlag für einen SW-Entwurf) entwickelt, die als Input in den *SW-Entwurf-Projekt* (als Vorgabe) hineinfliesst. Die Anforderungs-Definition wird unter Berücksichtigung der Usability- und SW-Engineering-Aspekte entwickelt.

Im folgenden werden alle Schritte der Entwicklung der Anforderungs-Definition exemplarisch beschrieben, die in dem UE-Prozess-Modell für die Entwicklung der Anforderungs-Definition wichtig sind. Zu Beginn jeder Abwicklung wird die Aktivität genannt, in der der Prozess gerade abläuft.

5.1 Erhebung von Kontext-Szenarien

Die Erhebung eines Kontext-Szenarios findet in der Aktivität Analytische Usabillity **(UA)** innerhalb der (Haupt-)Aktivität Usabilityanalyse **(U)** statt (Kapitel 4.2.1):

Im folgenden wird aus der Kenntnis der organisatorischen Abläufe einer Beratungsfirma ein Kontext-Szenario hergeleitet: In der System&SW-Beratungsfirma (SBF) gibt es verschiedene IT-Spezialisten. Die Aufgabe eines Beraters ist nicht nur, die Kunden in IT-Fragen zu beraten, sondern auch sich selbst ständig fortzubilden. Die Mitarbeiter sollen sich nicht nur im eigenen Themenbereich auf den neuesten Stand bringen, sondern auch von den anderen IT-Gebieten mindestens ein allgemeines Wissen erwerben, um so einen Überblick (Verstehen der Zusammenhänge) über verschiedene IT-Themenbereiche zu erlangen. Das Ziel der SBF ist also innerhalb der Firma das „Von-Einander-Lernen“ zu fördern. Zu diesem Anlass wurden bis jetzt sog. MA-Meetings (MA-Treffen) veranstaltet, um den Erfahrungsaustausch für alle Mitarbeiter (MA) innerhalb der SBF zu ermöglichen. Es werden dazu alle MA aus den drei Filialen eingeladen. Der IT-Austausch findet alle 3 Monate statt. Davor entscheiden die Projektleiter auf den Projektleitersitzungen, welche IT-Themen aus den Projekt-Erfahrungen vorgetragen werden sollen und welche MA vortragen sollen. Sie werden dann von den ausgewählten MA vortragstauglich vorbereitet - die zuständigen MA bereiten ihre Projekt-Erfahrungen und –Erkenntnisse in Form von Vortragsfolien jeglicher Art auf.

Alle erdenklichen Hilfsmittel zur digitalen Textverarbeitung und Wiedergabe, einschliesslich „Beamer“ stehen den Beratern bei der Vortragserstellung und Wiedergabe zur Verfügung, und sie haben keine Probleme bei dem Umgang mit Textverarbeitungssystemen.

Die organisatorischen Vorbereitungen werden durch die jeweiligen Sekretärinnen koordiniert. Sie schicken 1 Woche vor dem MA-Meeting als Erinnerung eine E-Mail an alle MA, in

der die Agenda des Meetings aufgelistet ist. Da die Meetings zeitlich begrenzt sind, können nur etwa 3 Vorträge a 1h/ pro Meeting gehalten werden (bei 200 MA), dabei ist aber das Interesse an verschiedenen IT-Themen viel grösser. Zusätzlich wird jeder der Vortragenden gebeten, seine Vortragsdokumente in digitalisierter Form (z. B. Word, PowerPoint usw.) in seinem Home-Verzeichnis aufzubewahren, um sie auf spätere Anfragen von Kollegen per E-Mail zusenden zu können; vorausgesetzt sie sind erreichbar.

Aus den o.g. Gründen wäre eine digitalisierte, immer verfügbare Form des IT-Erfahrungsaustausches unabhängig von den Meetings wünschenswert. Fazit: Es Bedarf möglicherweise einer technischen Ausstattung, wie sie von Entwicklern eines Intranet angeboten wird.

Für die Anforderungs-Entwicklung einer Intranet-Lösung wäre im Allgemeinen folgendes zu untersuchen: Ob und wie wird der Erfahrungsaustausch sw-technisch (SWP) unterstützt? Erleichtert die SW-Nutzung die Organisation der Meetings, die Vortragsvorbereitung und das Verstehen der Sachverhalte? Kann die Software schlecht gestaltete Arbeit verbessern helfen? Sind die Informationen, die zu Erledigung der Aufgaben notwendig sind, digital verfügbar? Unterstützt das SWP die für die Aufgabendurchführung erforderlichen Berechtigungen? Sind alle erforderlichen Hilfsmittel für die am SWP auszuführenden Tätigkeiten gegeben? Sind die Informationen, die zu Erledigung der Aufgaben notwendig sind, verfügbar und auf dem Bildschirm deutlich erkennbar? Werden Aufgaben, die das System selbst ausführen sollte, vom System automatisch ausgeführt?

Diese Beispiel-Anforderungen (um einige zu nennen) sind dann ein Teil des „Lasten-Heftes“.

An der Tätigkeitsaufgabe „Erfahrungsaustausch“ sind folgende Zielgruppen beteiligt:
- Projektleiter,
- Sekretärinnen,
- Vortragende,
- Interessenten: z.B. Herr Meier (Name zufällig gewählt)

Für diese Aufgabe müsste man ein Kontext-Szenario für jede Zielgruppe für ihre jeweiligen „key-tasks“ erstellen und analysieren, um so Anforderungen für jede Zielgruppe ableiten zu können. Die Analyse richtet sich dann vor allem nach den „key-tasks“ der einzelnen Zielgruppen.

Im Folgenden wird für eine Zielgruppe illustrativ ein Kontext-Szenario erstellt: Im Prinzip sind alle MA der SBF Zuhörer, aber eigentlich gemeint sind die Zuhörer, die an dem Erfahrungsaustausch (Treffen) nicht nur teilnehmen, sondern wirklich an dem Wissenstransfer interessiert sind - die Interessenten z. B. *Herr Meier*.

Leitfragen	**Kontext-Szenario**	**Aufgabenerfordernis**	**Dialogprinzip**	**Anforderungen an das Dialogsystem DS (Prüfkriterien für Lasten-Heft)**
Einleitung	***Einleitung***	***Einleitung***	***Einleitung***	***Einleitung***
1. Formulieren Sie die Tätigkeit in einem oder zwei Sätzen:	Hr. Meier will sich auf den MA-Treffen „Erfahrungsaustausch“ der SBF über die neuen Erkenntnisse in der IT-Branche informieren, um auf dem neuesten Stand zu sein.	Sind die Vortragsunterlagen verfügbar?	Aufgabenangemessenheit.	Sind die Vorträge am DS zugreifbar?
2. Aus welchen Aufgaben (key task) setzt sich die Tätigkeit zusammen?	1) Zum vorgegebenen Termin trifft Hr. Meier zu dem „Erfahrungsaustausch“-Treff in der SBF-Zentrale ein. 2) Hr. Meier nimmt an den Vorträgen teil, und beteiligt sich an den anschliessenden Diskussionen. Auf diese Art und Weise tauscht er mit allen IT-Erfahrungen aus.	1) Ist die Teilnahme an den Treffen immer möglich? 2) Kann man sich auf das Treffen vorbereiten? Sind die Diskussionsergebnisse verfügbar?	1) Aufgabenangemessenheit. 2) Steuerbarkeit.	1) Sind die Termine am DS rechtzeitig bekannt? 2) Sind die Vorträge vor und nach dem Treffen am DS zugreifbar? Kann man am DS vom Vortrag aus auf die Diskussionsergebnisse zugreifen?
3.Wie ist die Tätigkeit organisiert?	1) Der Erfahrungsaustausch findet alle 3 Monate an sog. MA-Treffen statt. Hr. Meier bekommt	1) Findet man auch den Termin, ab wann die	1) Selbstbeschreibungsfähigkeit, Steu-	1) Sind am DS die Vortragsunterlagen termin-

	die Termine von der Sekretärin. 2) Hr. Meier kann 3 Vorträgen pro Meeting a 1h zuhören und sich dann an der Diskussion beteiligen. 3) Bei Bedarf kann Hr. Meier die Vortragsunterlagen beim jeweiligen MA per Telefon oder E-Mail anfordern, da die Aufbewahrung der Vorträge in digitalisierter Form (z. B. Word, PowerPoint usw.) erfolgt.	Vortragsunterlagen verfügbar sind? 2) Sind alle Vortragsunterlagen verfügbar? 3) Können die Vortragsunterlagen auch unangefordert verfügbar sein?	erbarkeit. 2) Aufgabenangemessenheit. 3) Steuerbarkeit.	gerecht zugreifbar? Kündigt das DS den Eingang der Vortragsunterlagen an? 2) Sind die Termine am DS rechtzeitig bekannt? 3) Kündigt das DS den Eingang der Vortragsunterlagen an?
Voraussetzungen	***Voraussetzungen***	***Voraussetzungen***	***Voraussetzungen***	***Voraussetzungen***
4. Welche Qualifikation ist zur Bewältigung der Aufgaben erforderlich (Aufgabenbewältigung/ Softwarenutzung)? Welche Vorkenntnisse fehlen ggf.?	1) Notwendig sind Kenntnisse im IT-Bereich, um den Vorträgen das entsprechende abzugewinnen. 2) Der Umgang mit den E-Mail-Programmen, Textverarbeitungssystemen ist für die Beschaffung der Vortragsunterlagen notwendig.	1) Sind neben den Vortragsunterlagen andere Quellen verfügbar?	1) Aufgabenangemessenheit.	1) Sind im DS weitere Quellen kenntlich gemacht und zugreifbar?
5. Wer (bzw. welches Ereignis) bestimmt, was zu tun ist? (Wer trifft die Auswahl? Selbständigkeit der Bearbei-	Hr. Meier bekommt per E-Mail die Termine mitgeteilt und kann bei Bedarf Vortragsunterlagen anfordern.	Erfordernisse wie oben.		

tung, Arbeitsteilung, externe Datenquellen)				
6. Welche Hilfsmittel sind erforderlich (für die Aufgabenbewältigung/ zur Softwarenutzung)?Welche davon fehlen ggf., welche sind zusätzlich gewünscht?	Terminkalender.	Kann der Benutzer den Terminkalender selbst führen, oder/ und auch durch andere führen lassen?	Steuerbarkeit, Erwartungskonformität.	Ist der Terminkalender digital zugreifbar? Sind die Termineintragungen im Kalender zuverlässig? Kann man vom Terminkalender aus auf die Vortragsunterlagen zugreifen?
Normale Durchführung	***Normale Durchführung***	***Normale Durchführung***	***Normale Durchführung***	***Normale Durchführung***
7. Welche Arbeitsschritte sind durchzuführen?	1) Hr. Meier möchte sich auf den für ihn wichtigen Vortrag vorbereiten können. Er möchte immer die Termine rechtzeitig mitgeteilt bekommen. 2) Wann immer möglich nimmt Hr. Meier an den Treffen teil. 3) U. u. will Hr. Meier sein Wissen nach dem Treffen vertiefen, indem er danach die Vortragsunterlagen oder andere Quellen liest.	1) Erfordernisse wie oben. 2) Können die aktuellen Vortragsunterlagen während des Treffens ausgeliefert werden? 3) Erfordernisse wie oben.	2) Steuerbarkeit. Selbstbeschreibungsfähigkeit.	2) Sind am DS die aktualisierten Vortragsunterlagen zugreifbar? Ist am DS der Stand der Unterlagenbearbeitung erkennbar?

8. Welche Arbeitsschritte kehren häufig wieder? (Automatisierung gewünscht/ erforderlich?)	Hr. Meier nimmt seine Unterlagen ausgedruckt mit, weil er sie während der Reise in der Bahn gerne liest.	Werden die Unterlagen im druckfähigen und gut lesbaren Zustand (formatiert) angeboten?	Aufgabenangemessenheit, Erwartungskonformität.	Kann man am DS Unterlagen ausdrucken?
9. Welche Arbeitsschritte werden automatisch durchgeführt? Sind bei automatisierten Arbeitsschritten Einflussmöglichkeiten des Benutzers vorhanden/ erlaubt/ gewünscht/ erforderlich?	1) Hr. Meier kümmert sich um alles, was mit der Vorbereitung und Teilnahme an dem Treffen zu tun hat, selbst. 2) Hr. Meier wünscht sich, entlastet zu werden.	1) Sind die erforderlichen Termine und Unterlagen ohne Probleme auffindbar? 2) Kann die Terminüberwachung automatisiert werden?	Steuerbarkeit.	1) Anforderungen wie oben. 2) Hat das DS eine integrierte Kalenderfunktion?
10. Kommt es vor, dass mehrere Benutzer gleichzeitig an dem gleichen Objekt (z.B. Vorgang, Akte, Dokument, Datensatz) arbeiten müssen?	Informationen, die auf den MA-Treffen ausgetauscht werden, sind ausschliesslich für die MA des SBF. D.h. nur die MA haben darauf Zugriffsrechte. Hr. Meier kann, wie alle anderen MA alle Vortragsdokumente anfordern, die schreibgeschützt sind.	Sind Zugriffsrechte für alle definiert?	Erwartungskonformität.	Sind alle zugreifbaren Vortragsunterlagen und Diskussionsobjekte unmittelbar verfügbar?
11. Gibt es eine festgelegte Abfolge der Arbeitsschritte und wenn ja, wie sieht diese aus? (Ist eine Flexibilität sinnvoll/ nötig?)	Hr. Meier ist festgelegt in der Beachtung der Termine, aber es ist ihm freigestellt, ob er sich auf die Vorträge vorbereiten will.	Werden die Termine rechtzeitig in Erinnerung gebracht?	Erwartungskonformität.	Erinnert das DS an festgelegte Termine?
12. Welche (Teil-) Ergebnisse entstehen und wie werden diese ggf. verwertet/ weitergeführt?	Manchmal braucht Hr. Meier Folien von den Vorträgen.	Sind Folien der Vorträge zentral verfügbar?	Steuerbarkeit.	Sind Folien im DS an einer Stelle abgelegt und zugreifbar?
13. Welches Feedback be-	Entfällt.			

kommt die befragte Person in bezug auf die Arbeitsergebnisse und die Wirkung ihrer Arbeit?				
Besonderheiten bei der Durchführung	***Besonderheiten bei der Durchführung***	***Besonderheiten bei der Durchführung***	***Besonderheiten bei der Durchführung***	***Besonderheiten bei der Durchführung***
14. Welche Unterbrechungen gibt es und warum? Welche Störungen treten auf (organisatorisch/ sozial/ technisch)?	1) Hr. Meier kann wegen Dienstreisen, Urlaub, Krankheit u. ä. verhindert sein, und nicht an dem Erfahrungsaustausch teilnehmen. 2) Es kann passieren, dass ein Vortrag oder sogar das ganze MA-Treffen ausfällt. 3) Es kann passieren, dass der Vortragende aus bestimmten Gründen nicht erreichbar ist, so dass Hr. Meier die notwendigen, sogar dringlichen Vortragsunterlagen erst mit Verspätung bekommt.	1) und 3) Sind die Vortragsunterlagen vor und nach den Treffen verfügbar? (Erfordernisse wie oben) 2) Wird der Termin bekannt gegeben und ein Ersatztermin angegeben?	2) Erwartungskonformität.	2) Kann das DS Änderungen besonders hervorheben?
15. Wie werden Fehler zurückgemeldet und behoben (organisatorisch/ sozial/ technisch)?	Es ist schon mal vorgekommen, dass zu einem Termin falsche Unterlagen versendet wurden.	Ist falsche Zuordnung korrigierbar?	Fehlertoleranz.	Erlaubt das DS die Korrektur der Zuordnung?

16. Welche wichtigen Sonderfälle müssen berücksichtigt werden (bzw. fallen dem Benutzer spontan ein; z. B. zur Arbeitsteilung/ Zusammenarbeit)?	Es gibt keine Sonderfälle.			
Organisatorische Rahmenbedingungen	***Organisatorische Rahmenbedingungen***	***Organisatorische Rahmenbedingungen***	***Organisatorische Rahmenbedingungen***	***Organisatorische Rahmenbedingungen***
17. Welche Organisationsziele gibt es im Hinblick auf die Tätigkeit?	Das Ziel der SBF ist es, innerhalb der Firma das „Von-Einander-Lernen“ zu fördern.	Alle Lernmittel sind allen verfügbar?	Aufgabenangemessenheit.	Wird das Ziel der Org. durch das DS ausreichend unterstützt?
18. Gibt es Mechanismen zur Leistungssteuerung/ Leistungskontrolle? (Wenn ja, welche? Sind diese erforderlich?)	Es gibt keine Anwesenheitsliste und keine Leistungskontrolle.			
19. Welchen Überblick hat der Benutzer im Hinblick auf die Gesamttätigkeit?	Hr. Meier hat eine Übersicht über Themen, die für das laufende Jahr geplant sind.	Ist jemand für die Planung der Treffen zuständig?	Erwartungskonformität.	Ist in DS ein Plan zugreifbar?
20. Von welchen Arbeitsergebnissen/ Arbeitsschritten sind Dritte (z. B. Kunden) direkt betroffen? Und was folgt daraus?	Wenn Hr. Meier unbedingt am MA-Treffen teilnehmen will, muss er Kundentermine verschieben oder absagen.	Ist die Möglichkeit zur autonomen Terminplanung gegeben?	Steuerbarkeit.	Kann das DS rechtzeitig aktualisierte Termine anbieten?
21. Welche Änderungen, die die Aufgabenbearbeitung beeinflussen, sind zu erwarten oder werden gewünscht? Welche Vorschläge hat der/die Befragte dazu?	Es ist geplant, dass die Treffen an verschiedenen Standorten der Firma angeboten werden, damit die Reisebelastung der MA besser verteilt wird.	Ist die Ausstattung der Filialen für die Durchführung von Treffen ausreichend?	Steuerbarkeit.	Kann das DS an die Änderungen angepasst werden?

22. Welche Stressfaktoren gibt es und wie wird damit umgegangen?	Entfällt.			
Sonstige Bemerkungen der befragten Person zu bereits aufgetretenen Nutzungsproblemen.	Die jetzige Form des Erfahrungsaustausches verursacht Anfahrtskosten, erreicht nicht alle MA, was die Idee der Veranstaltung verfehlt – es belastet im engeren Sinne den MA enorm. Die MA, die sich aus bestimmten Gründen nicht informieren/austauschen können, müssen sich das Wissen woanders beschaffen, was die SBF zusätzlich Zeit und Geld kostet.	Bestehen Möglichkeiten (Vorschläge), die Nutzungs-Probleme zu lösen? Kann eine veränderte Gestaltung des Nutzungs-Kontextes die Nutzungs-Probleme lösen?	Organisations-Entwicklung.	

Das Kontext-Szenario kann auch noch um andere Details erweitert werden, wie Bereiche, Randgebiete, Nachbarschaftswelten, Mittel, Dokumentationsformen, Regelwerke, Internet, virtuelle Umgebung, die relevant für die valide Nutzungs-Beschreibung (***UNB)*** sind. In diesem Fall wäre eine Einigung z. B. über eine einheitliche Dokumentationsform nötig.

Bevor man mit der ***(UNB)*** in die Aktivität **(UE-D)** wechselt, muss sie in den Bereich des **(SM)** weiter geleitet werden. Also, die ***(UNB)*** fliesst als Input in die Aktivität (**SM-A)** ein. Anhand der dort beschriebenen Vorgaben wird das IT-Team sich eine System-Idee ***(SI)*** überlegen.

Die *Prüfkriterien **(PK.1)***, die aus dem *Kontext-Szenario **(UKS)*** abgeleitet werden, können als Bestandteil des Lasten-Heftes verstanden werden, gegen die man die Merkmale des SWP prüft, z. B.: „Ist eine bestimmte Rückmeldung des SWP zu dem gegebenen Erfordernis angemessen?“, oder genauer „sind die Vortragsunterlagen für alle MA am Dialogsystem zugreifbar?“. Wichtig ist bei der Formulierung des Prüfkriteriums, dass man es im nachhinein (auch ohne Benutzerbeteiligung) mit einem klaren „Ja“ oder „Nein“ beantworten kann.

Für einen Projekt-Plan **(PPL),** der hier generiert wird, sind z. B. folgende Aspekte wichtig: Zusammenstellen eines Usability- und IT-Teams (siehe Kapitel 4), falls notwendig einen Moderator zur Verfügung stellen, Erstellen eines allgemeinen Projekt-Plans (nach ISO 13407-Pkt.6).

5.2 Entstehung der System-Idee

In der Aktivität System-Machbarkeitsanalyse **(SM-A)** entsteht die *System-Idee **(SI)*** (Kapitel 4.2.3):

Nach der Analyse der ***(UNB)*** anhand von **SE.1** stellte das IT-Team fest, dass es einer Intranet-Lösung bedarf. Das Team wählte zur Diskussion für die Aktivität **(UE-D)** „BSCW“, „LotusNotes“ und „Zeno“ als eine ***(SI)*** aus, einige am Markt zur Verfügung stehenden CSCW-Anwendungen.

Die Hauptaufgabe hier ist aus der ***(UNB)*** Sicht, die Kernfunktionalität zu beschreiben, und den Funktionsumfang im allgemeinen Sinne anzugeben, und zwar in Form eines „Lasten- und Pflichten-Heftes“. Das Ergebnis sind Anforderungen **(UE-A)**, auf die man sich geeinigt hat, die als Vorgabe für **(UK)** und **(SM-E)** dienen.

Für die sw-technische Entscheidungsfindung ist das Vorgehen nach **SE.3** sehr hilfreich. Hier werden die Unterschiede der technischen Ausstattungen von BSCW, LotusNotes und Zeno analysiert, verglichen, und fliessen als Vorschlag ebenfalls in die **(UE-D)** ein.

5.3 Abbau der Dialektischen Barriere

In dem Dialektischen Prozess **(UE-D)** wird man Möglichkeiten abwägen (BSCW, LotusNotes oder Zeno), Widersprüche aufdecken (nach der Formel: Man macht einen Vorschlag, dieser Vorschlag wird auf Widersprüche zum erwarteten oder gewünschten Nutzungskonzept geprüft, die Widersprüche wird man dann zu beseitigen versuchen, und dann den Vorschlag erneut prüfen.), um zu ersten, validierten Anforderungen zu gelangen (Kapitel 4.2.2):

Man begibt sich in den Dialektischen Prozess, um Ziele („Was wollen wir eigentlich erreichen?“) zu klären und die Mittel zu präzisieren („Womit können wir es erreichen?“)

Die beiden Teams stellten fest, dass alle angebotenen Systeme nur aus der Sicht des Anbieters entwickelt wurden und man sie nicht ohne weiteres im gegebenen Nutzungs-Kontext verwenden kann.

Man einigt sich auf „Zeno“ (als Zeichen der Konsensfindung ***(UE-KP)***), weil man es am einfachsten für die schon geklärten Anforderungen der Firma modifizieren kann. Das „Zeno“-System betrachtet man als einen Prototypen, den man in **(UE-S)** mit Benutzerbeteiligung ausprobieren wird. Bevor dies allerdings geschieht, wird „Zeno“ in **(UK)** auf **critical incidents** untersucht.

5.4 Erstellung eines Use-Szenarios

Die Aufgabe der Erstellung von Use-Szenarien hat die Aktivität Konstruktive Usability **(UK)** (Kapitel 4.2.1):

Aus dem Nutzungs-Kontext ***(UNK)*** haben sich u. a. folgende Aufgaben (key-task) herauskristallisiert, wie: das „Einloggen“, Bedienung der „Befehl-Navigation“, das „Ablegen“, „Holen“ von Informationen (Vortragsunterlagen), von Daten und der „Feedback“. Für all diese Aufgaben wird dann ein Use-Szenario erstellt, dass ähnlich zu den Prinzipien der Use-Cases erfolgt (genauer im Kapitel 2).

Im Folgenden wird anhand einer Aufgabe, nämlich „Holen“, demonstriert, wie ein Benutzer sich die Informationsunterlagen aus Zeno „holen“ kann.

Da Zeno für die Nutzung erprobt wird, richtet sich der Blick bei den Ausschnitten der Aufgabe in **(UC)** vor allem auf die **critical incidents.**

Vorerst aber drei Abbildungen zur optischen Problemdarstellung der Arbeitsbereiche auf dem Bildschirm.

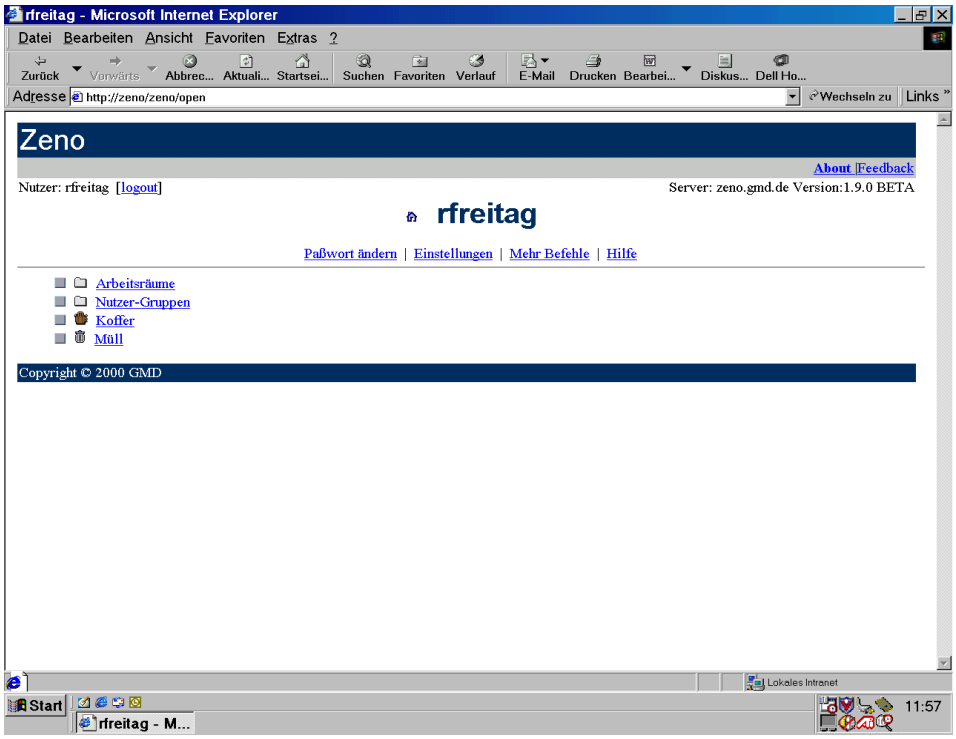

Abbildung 5-1 Beispiel 1 [URL:ZENO]

Nach dem sich Herr Meier in sein Home-Verzeichnis, wo ihm die Grundfunktionen zur Verfügung stehen, eingeloggt hat, wählt er die Funktion „Arbeitsräume“ aus, und es wird ihm eine Auflistung aller Arbeitsräume präsentiert.

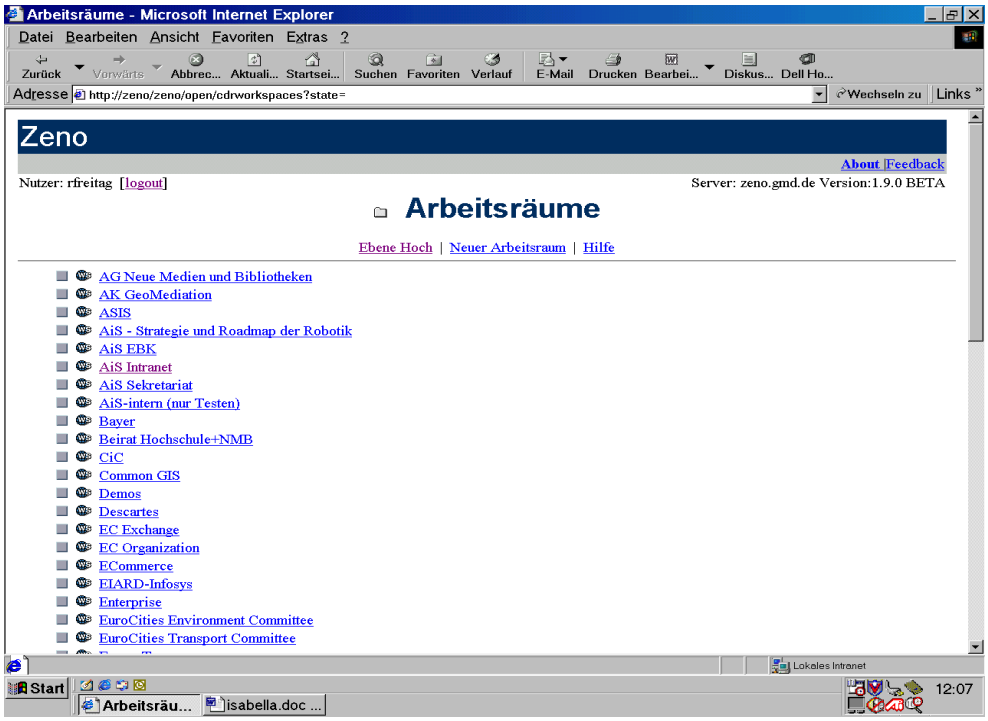

Abbildung 5-2 Beispiel 2 [URL:ZENO]

Es werden zwei wesentliche Aspekte dieser Aufgabe „Holen“ analysiert, nämlich Herr Meier sucht nach Unterlagen mit einem bestimmten Titel aus der Abb. 5-2 , oder er will in einem Arbeitsbereich nachschauen, welche Titel es bereits schon zu einem bestimmten IT-Bereich des Arbeitsbereichs gibt. Das letztere Ergebnis ist in Abb. 5-3. dargestellt.

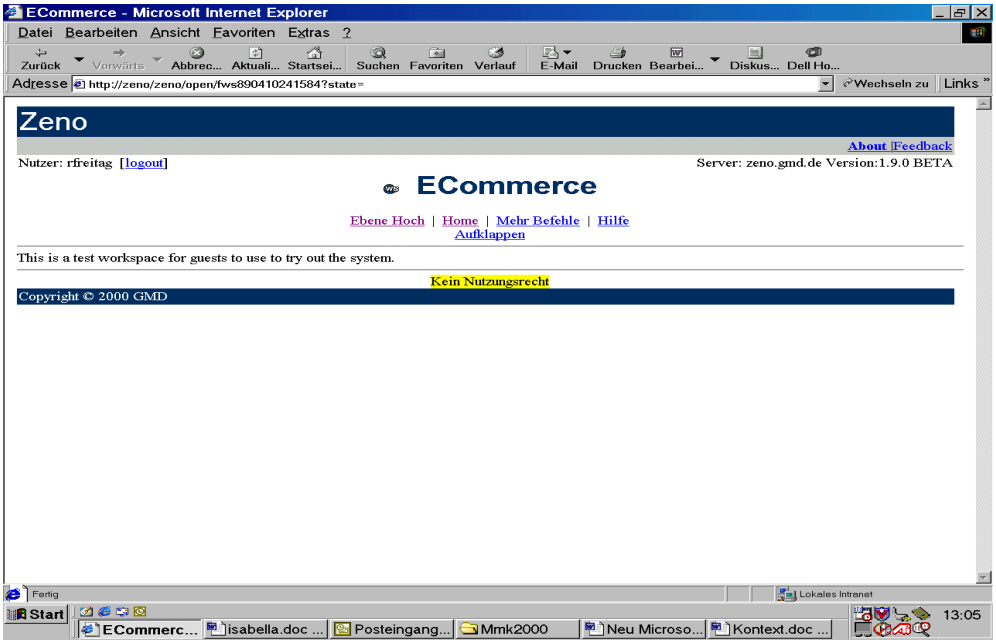

Abbildung 5-3 Beispiel 3[URL:ZENO]

Wie man in der Abbildung 5-3 sieht, erfährt Herr Meier erst auf der ausgewählten Arbeitsbereichs-Ebene, dass er auf sie keinen Zugriff hat. Damit ist seine Erwartung nicht erfüllt, dort Titel zu dem ausgewählten IT-Bereich vorzufinden. Denn an Zugriffsinformation ist er auf dieser Ebene nicht interessiert. Dies stellt ein Problem für den Benutzer dar, weil es ihn hindert, seine Aufgabe effizient zu erledigen.

Aufgabe in Teilaufgaben zerlegen	***Use-Case***		**Anforderungen oder zu klärende Fragen**
	Benutzer-Aktion	**Aktion/Reaktion des Dialogsystems**	
1. Teilaufgabe: Finden von Informationsunterlagen mit bestimmtem Titel	Hr. Meier klickt Icon „Arbeitsräume“ an.	Die Anwendung (SWP) öffnet ein neues Dialogfenster. **critical incident: Das Dialogfenster ist nicht zur Suche von Informationsunterlagen mit bestimmten Titel vorgesehen, sondern listet nur die allgemeinen IT-Bereiche auf.**	Das neue geöffnete Dialogfenster muss eine Such-Funktion anbieten, damit man einen Titel finden kann. (ISO 9241-10 „Die Form der Eingabe und Ausgabe sollte der jeweiligen Arbeitsaufgabe und den Benutzerbelangen angepasst sein.)

2. Teilaufgabe: Finden einer Auflistung der Titel aus einem bestimmten IT-Bereich	Meier klickt Icon „eCommerce“, weil er nach einer Verzeichnisliste sucht, um sich zu erkundigen, welche Titel zu dem ausgewählten IT-Bereich (eCommerce) bereits existieren, die er sich dann durchlesen könnte.	Eine Meldung erscheint: Die Anwendung (SWP) informiert Herrn Meier nur darüber, dass er für den Arbeitsbereich „e-Commerce“ keinen ausreichenden Zugriff hat. **Critical incident: Die Erwartung von Herr Meier, hier eine Titelauflistung zum Thema „eCommerce“ vorzufinden, geht nicht in Erfüllung.** **Critical incident: Meier versteht an dieser Stelle die Meldung nicht.** **Es werden auch keine Vorschläge gemacht, was zu tun ist, um einen Zugriff zu ermöglichen.**	Nach dem Klick auf einen Arbeitsbereich (IT-Bereich) muss der Inhalt (IT-Themen) des Bereiches aufgelistet werden. Die Zugriffsrechte müssen vorher schon geklärt worden sein. Vorab aber ist eine Frage noch zu klären, ob die Firma SBF überhaupt die Zugriffspolitik für das Zeno braucht. Es ist bekannt, dass alle MA sich informieren sollen und dass Zeno eine Intranet-Lösung ist, d.h. die Zugriffsvergabe ist ohnehin nur innerhalb der MA der Firma erlaubt. ISO 9241-10 „Rückmeldungen oder Erläuterungen sollten genau auf die Situation bezogen sein, für die sie gebraucht werden, um ihren Wert für den Benutzer zu erhöhen.
	Meier versucht jetzt nacheinander alle „Arbeitsbereiche“ auszuwählen, um herauszufinden, in welchen „Arbeitsbereich“ er überhaupt was zum IT-Thema finden kann.	Die Anwendung (SWP) liefert zu 90% die bekannte Meldung: „ungenügender Zugriff “. Die Anwendung (SWP) liefert in 10% **Critical incident: Eine Auflistung der Arbeitsbereiche auf die Herr Meier Zugriff hat.**	Nach dem Klick auf einen Arbeitsbereich (IT-Bereich) muss der Inhalt (IT-Themen) des Bereiches aufgelistet werden.

Abbildung 5-4Ein UC-Beispiel für Zeno

Man kann sagen, dass die Nutzung der Arbeitsbereiche den erhobenen Anforderungen der SBF nicht entspricht, so dass sich in diesem Fall aus den **critical incidents** keine Einarbeitungsprobleme ableiten lassen, sondern *Nutzungsprobleme*.

Das Dokument ***(US)*** geht als Produktdokument in die Weiterentwicklung (Verbesserung) des Prototypen ein.

Diese Erkenntnisse aus der Nutzung des Prototypen liefern die Modifizierungsvorgabe, für die der Zeno-Prototyp geändert werden muss, um für die Benutzer gebrauchstauglich zu werden.

5.5 System-Machbarkeitsentwurf

In der Aktivität System-Machbarkeitsentwurf **(SM-E)** wird ein Lösungs-Vorschlag ***(LV)*** entwickelt (Kapitel 4.2.3):

Die Aufgabe liegt im Anpassen (Modifizieren) der Zeno-Intranetlösung an Anforderungen (Bedürfnisse) der Firma SBF, damit man Vortragsunterlagen durch selbstständiges Suchen/ Finden/ Ablegen von Informationsunterlagen zu den IT-Bereichen/ Themen durchführen kann. Auch ein Diskussionsforum (mit Feedback) soll ermöglicht werden.

Nach der Analyse der ***(UE-A)*** und anhand von **SE2** entwirft das IT-Team Zeno unter diesen Aspekten und schlägt es als einen Lösungs-Vorschlag ***(LV)*** für das Explorative Prototyping **(UE-EP)** für die Aktivität **(UE-S)** vor.

Die Hauptaufgabe hier ist, auf der Basis der *Validierten Anforderungen* ***(UE-A)*** einen Lösungs-Vorschlag ***(LV)*** für eine mögliche funktionale Struktur des zukünftigen SWP zu erarbeiten. Z. B. ist gefordert, dass alle MA der Firma - dieses Intranet wird nur in dieser Firma genutzt - sich informieren können sollen, deshalb werden die Funktionalitäten „Login“ und „Passwort“ in dem zukünftigen Zeno nicht mehr gebraucht und können aus dem jetzigen Zeno sw-technisch entfernt werden.

Für die sw-technische Entscheidungsfindung ist das Vorgehen nach **SE4** sehr hilfreich. Hier kann die technische Ausstattung von Zeno analysiert und ausgearbeitet werden und als Vorschlag ebenfalls in die **(UE-S)** einfliessen.

5.6 Abbau der Synthesebarriere

In dem Synthese-Prozess **(UE-S)** wird der Zeno-Prototyp in dem sog. **(UE-EP)** erprobt (Kapitel 4.2.2):

Man begibt sich in den Synthese-Prozess, um das Ziel (Zeno für die o. g. Aufgabe der Firma nutzbar zu machen) zu erreichen. Dafür werden hier die Funktionen von Zeno (Mittel) auf Widersprüche zu den Nutzungs-Anforderungen untersucht und ggf. angepasst (behoben).

Der aus **(SM-E)** vorgeschlagene Zeno-Prototyp ***(LV)*** wird anhand der ***(US)*** Vorgaben getestet. Die Ergebnisse bilden dann einen T-Prototypen, dessen Merkmale anschliessend evaluiert werden **(UE-ME)**. Daraus geht hervor, dass man z. B. keine Zugriffsrechte auf die „Arbeitsbereiche“ mehr braucht, so dass beim Einklicken weder eine Meldung über Zugriffsrechte, noch die Auflistung der zugriffsfreien „Arbeitsbereiche“ notwendig sind, sondern an dessen Stelle kommt die gewünschte Auflistung der IT-Themen, über die sich alle MA informieren können. Nach dem Klick auf die Inhalte werden diese präsentiert.

Ausserdem werden bei der Merkmal-Evaluierung **(UE-ME)** auch Prüfkriterien ***(PK.3)*** bzgl. der Merkmale abgeleitet. Zum Beispiel „Sind die Vortragsunterlagen für alle MA am Dialogsystem zugreifbar?“

Nachfolgend ein Beispiel für eine *Claims Analysi*s im Rahmen des Synthese-Prozesses:

Nach dem Abgleich der Use-Szenarien mit der System-Machbarkeitsanalyse finden die beiden Teams ein Merkmals-Problem **(UE-MP)**, welches die Klärung der Zugriffsrechte der MA auf das Intranet betrifft. Dazu wird in **(UE-R)** eine „Pro und Contra -Tabelle“ (genauer im Kapitel 2.2.4) aufgestellt und ausgewertet, um die kontextbezogenen Vor- und Nachteile zu analysieren. Dazu konsultiert man auch die sw-technischen Möglichkeiten aus **(SM-F)**. Man führt eine Umfrage bei Benutzern (MA der Firma) durch und stellt fest, dass drei Argumente für den Firmenzugriff immer wieder bestätigt wurden und kaum eindeutige Argumente für einen persönlichen Zugriff. Die Schlussfolgerung aus dem Dialektischen Prozess lautet: Die Idee ist doch, dass alle MA sich informieren sollen, und weil das Zeno eine Intranet-Lösung ist, d.h. die Zugriffsvergabe ist nur innerhalb der MA der Firma erlaubt, sollen alle MA uneingeschränkten Zugriff auf Zeno haben. Ferner: Jeder MA ist selbst für seine Meinung verantwortlich, die er im Diskussionsforum äussert. Zusätzlich kommt die Firmenphilosophie hinzu, die besagt: Respektvoller Umgang mit MA und Transparenz der firmeninternen Handlungen. Ausserdem ist es system-technisch machbar und ohne Mehrkosten durchführbar.

Die *Merkmal-Änderung **(MÄ)*** „Verzicht auf Passwörter beim Zugriff auf Zeno-Dokumente“ wird zu den *Validierten Anforderungen **(UE-A)*** aufgenommen, um noch einmal gemeinsam mit den anderen Anforderungen in den Synthese-Prozess **(UE-S)** einzugehen, wo sie ausprobiert werden, falls den Benutzern noch etwas unklar erscheint.

5.7 Pflege-Prozess

Der Pflege-Prozess **(UE-Pfl)** wurde im Kapitel 4 und 4.2.4 ausführlich vorgestellt:

Um die neuen Nutzungs-Anforderungen *Qualitätserfordernisse **(QAE)*** ableiten zu können, wird man nach der Installation und Erprobung des SWP im Nutzungs-Kontext feststellen können, welche Fehler, Mängel auftreten, oder ob sogar der Nutzungs-Kontext selbst einer Weiterentwicklung (Organisations-Entwicklung **(VOE)**) bedarf.

Fazit: Mit der Erfahrung, die man durch die reale Nutzung des SWP im Kontext gewinnt, kann man die sw-technische und die Nutzungs-Qualität des SWP für die nächste Version verbessern, vorausgesetzt, man hat von Anfang an die Erkenntnisse aus dem Usability-Engineering in das gesamte Projekt einfliessen lassen.

6 Ausblick

Die Informationstechnik (IT) hat zum Ausgang des 20. Jahrhunderts viele Menschen zu Benutzern von Computern und Anwendungen gemacht. Das setzt aber nicht voraus, dass alle Benutzer immer tiefe, detaillierte Kenntnisse der sw-technischen Gegebenheiten haben müssen, um sich der Computerwelt bedienen zu dürfen; ganz im Gegenteil, die Computerwelt soll den Benutzern dienen. Deswegen muss sie so konzipiert sein, dass Benutzer keinen unnötigen Hindernissen ausgesetzt sind, um ihre Aufgaben effizient und auch mit Spass erledigen zu können. Die verbreitete Unzufriedenheit der heutigen Nutzer mit IT-Produkten zeigt deutlich, dass es an der Zeit ist, neue Wege zu gehen, um den Wünschen der Benutzer (Kunden) besser gerecht zu werden.

Die Gebrauchstauglichkeit der neuen Informationstechnik wird künftig zu einem zentralen Qualitätsmerkmal. Es werden nur die Geräte und SW-Produkte (SWP) gefragt sein, die gebrauchstauglich für die zu erledigenden Aufgaben sind, d. h. benutzbar, intuitiv, verständlich, mit geringer Ablenkung von der Aufgabe, nutzbar und ansprechend in der Ästhetik sind, so dass Spass an der Nutzung entsteht.

Die Kaufmotivation der vergangenen Tage - wie z. B. ein Unternehmen kauft ein SWP, weil es funktioniert oder die Konkurrenz auch schon damit arbeitet - wird künftig nicht mehr ausreichen, weil die Produktivität der Nutzung nicht nur durch die sw-technische Qualität, sondern vor allem durch die Nutzungs-Qualität gesteuert wird. Hersteller (Usability-Teams, IT-Berater), die den Nutzungs-Kontext eines zu entwickelnden SW-Produkts analysiert haben, können bereits anhand des Prototypen und mit Benutzerbeteiligung voraussagen, ob und inwieweit sich die Produktivität der späteren Benutzer verbessern lässt, was man auch in Zahlen ausdrücken kann. Ausserdem kann darüber hinaus die Mindestqualität bezüglich der ISO-Normen festgestellt werden. Ein Umdenken bei den Anwendern, was die Softwarenutzung betrifft, und die Forderung nach Einhaltung ergonomischer Normen soll die Hersteller veranlassen, Software zu produzieren, die wenigstens das Mindestniveau an Gebrauchstauglichkeit erreicht.

Mit der Steigerung der Gebrauchstauglichkeit steigt auch die Qualität des SWP, weil die Nutzungskosten sinken, was wiederum für ein Unternehmen Wettbewerbsvorteile mit sich bringt. Die Wettbewerbsvorteile so „ausgezeichneter“ SWP sind dem Laien noch nicht klar, aber einem gut ausgebildeten und erfahrenen IT-Manager schon längst vorstellbar.

Die IT-Branche wird zum Beginn des 21. Jahrhunderts erst durch die Verbesserung der SW-Entwicklungs-Prozesse zeigen müssen, dass sie im Stande ist, benutzerorientierte SW-Entwicklung zu gestalten, um den Anwendern (Unternehmen) qualitative SWP anzubieten und das Vertrauen der Benutzer wieder zu gewinnen.

In einer Studie des Forschungsverbundes SW-Engineering (Informatik Spektrum 22/1999: Seite 24-36) wurden durch Interviews die derzeit praktizierten SW-Entwicklungs-Prozesse untersucht. Unter anderen lag der Forschungsschwerpunkt auf der Anwendermodellierung unter Einbeziehung der Benutzer. Im Fazit der Untersuchung schreiben die Autoren u. a.: “Neben diesen technischen Kernfragen..., ergeben sich auch weitere, eher sozial orientierte Problemstellungen, die nicht in die Thematik des Forschungsverbundes gehören, aber dennoch wichtig sind. So hat sich gezeigt, dass der Informationsfluss zwischen den an der SW-Entwicklung beteiligten Personen verbessert werden muss. Der mangelnde Wissensaustausch rührt in erster Linie daher, dass die Arbeiten und Leistungen der beteiligten Gruppen untereinander teilweise keine Wertschätzung finden... . SW-Entwickler hinterfragen selten wirt-

schaftliche Aspekte der SW-Erstellung und Manager wiederum haben oft nicht die nötige Ausbildung, um komplexe technische Zusammenhänge erkennen zu können... . Ein Beitrag zur Lösung dieser Problematik wäre möglicherweise, Informatiker nicht nur in Fachabteilungen, sondern auch vermehrt für Stabs- und Managementtätigkeiten einzusetzen um so die Kommunikation“ zu verbessern.

Eine ganzheitliche Vision für die zukünftige SW-Entwicklung ist notwendig. Integrierte Gestaltungsansätze des SW- und Usability-Engineering werden künftig die Lösung sein. Da das Entwicklungsfeld ständig wachsende Komplexität aufweist, wird keine Einzelperson in der Lage sein, einen SW-Entwicklungs-Prozess allein zu gestalten, und keine Einzeldisziplin ist in der Lage, dieses Aufgabenfeld allein zu bewältigen. Meist werden fähige Spezialistenteams (z. B. aus den Bereichen Informatik, Usability- Engineering, Psychologie, Soziologie, Grafikdesign, Arbeitsergonomie, Organisationswesen) einzusetzen sein, die in der Lage sind, miteinander zu kooperieren. Dabei muss die Nutzungs-Qualität mindestens denselben Stellenwert haben wie die sw-technische Qualität.

Software-Engineering und Usability-Engineering müssen einhergehen, damit ein kostengünstigerer SW-Entwicklungs-Prozess durch frühzeitiges Erkennen von Mängeln und eine fachgerechte Standardisierung des Vorgehens gewährleistet werden kann.

Die Umsetzung dieser Vorhaben verlangt in erster Linie nach Bildungsmöglichkeiten für Usability-Engineering. Diese Disziplin wird in der traditionellen universitären Informatik-Ausbildung kaum oder gar nicht angeboten. Gut ausgebildete Usability-Analytiker /-Engineer /-Prüfer sind aber für die Entwicklung von Nutzungs-Qualität unabdingbar.

7 Glossar

Die im folgenden Glossar aufgelistete Begriffe und ihre Definitionen stammen hauptsächlich aus den DATech-Büchern [DAT-G´00], [DAT-U´00]:

Abweichung (nonfulfilment/ departure): Unterschied zwischen einem gegebenen Produktmerkmal oder einer am Dialogsystem ausführbaren >Arbeitstätigkeit und einem >Prüfkriterium. Siehe auch Normabweichung, Fehler, Mangel.

Anforderung (requirement) auch Nutzungsanforderung: Ein qualitativer oder quantitativer Wert einer Qualitätsforderung, die aus >Erfordernissen der Arbeit oder Organisation (>Nutzungs-Kontext) des Benutzers sowie seiner Belange abgeleitet ist. Anforderungen sind i.d.R. als geforderte >Arbeitstätigkeit zu formulieren. Nur wenn gesicherte ergonomische Erkenntnisse für die Merkmale eines Produkts (z.B. Farbkodierung, Hintergrundfarbe des Bildschirms) gegeben sind, können diese Merkmale als Anforderung dienen. Eine Anforderung kann in ein >Prüfkriterium transformiert werden, wenn die Tätigkeitsforderung für die Interpretation (Präzisierung) einer genormten Anforderung [ISO-9241-10] benutzt wird.

Anwender (customer/ user organization): Eine Organisation bzw. Institution, die Rechensysteme zur Erfüllung von Datenverarbeitungs-Aaufgaben bzw. zur Unterstützung von Informationsverarbeitungs-Prozessen einsetzt. Hierbei ist es unerheblich, welchen Umfang die Rechneranwendung hat. Der Begriff Anwender wird bewusst nicht auf eine einzelne Person oder auf eine Personengruppe bezogen, die Rechner für ihre speziellen Aufgaben einsetzt. In letzterem Fall wird der Begriff >Benutzer verwendet. Die Begriffe Anwender und Benutzer werden nicht synonym verwendet; bei einem Anwender gibt es üblicherweise viele verschiedene Benutzer.

Arbeitstätigkeit (work activity): Eine beobachtbare Tätigkeit des >Benutzers am Bildschirmarbeitsplatz, die zur Erledigung der Arbeitsaufgabe ausgeführt wird. Nicht beobachtbar sind kognitive Prozesse wie lernen, erinnern, eine Vorstellung entwickeln (mentales Modell), denken, suchen usw.

Assessment: Eine Methode der Bewertung eines Prozesses zur Software-Erstellung beim CMM.

Benutzbarkeit (usability): Eine Menge von Merkmalen zur Erleichterung der Benutzung mit Rücksicht auf die individuelle Bewertung durch eine festgelegte oder vorausgesetzte Gruppe von >Benutzern. Benutzer (user/ end user) Eine Person oder eine Personengruppe, die Rechner zur Unterstützung ihrer Aufgabenbearbeitung einsetzt.

Benutzungsoberfläche (user interface): Der Begriff "Benutzungsoberfläche" hat den früher benutzten missverständlichen Begriff "Benutzeroberfläche" abgelöst. Bezeichnung für die Gesamtheit der Eingabemittel und Anzeigeeinrichtungen eines Systems (z. B. Tastatur, Cursor, Bildschirm, Anzeigen). Die Benutzungsoberfläche ist Teil der >Benutzungsschnittstelle.

Benutzungsschnittstelle (user interface): Gesamtheit der Eingabemerkmale, der Ausgabemerkmale, der Dialogmerkmale und der Werkzeugmerkmale, mittels derer der >Benutzer auf das System einwirkt (z.B. Eingabefeld, Kommandozeile, Unterbrechungskommando, UNDO) sowie der Merkmale, in denen sich das Verhalten des Systems gegenüber dem Benutzer äussert (z.B. Meldungen, Nutzungshinweise, Menüs am Bildschirm, nutzungsrelevante Werkzeugeigenschaften).

Design-Use-Cycle: Die Phasen, die ein Softwareprodukt durchläuft, beginnend mit der Produktidee über die Entwicklung des Nutzungskonzepts, über den Entwurf, die Implementierung und die Installierung des Produkts beim Anwender, bis hin zur >Pflege des Produkts im Nutzungs-Kontext, in dem aufgrund der Nutzungserfahrungen das nächste Release vorberei-

tet wird. Eine iterative Folge der Phasen ist üblich. Entwicklungsaktivitäten und Massnahmen der >Qualitätssicherung wechseln sich ab.

Dialog (dialogue): Der Dialog ist ein Ablauf, bei dem der Benutzer zur Abwicklung einer Arbeitsaufgabe - in einem oder mehreren Schritten - Daten eingibt und jeweils Rückmeldung über die Verarbeitung dieser Daten erhält (DIN-66 234, Teil 8).

Dialogschritt (dialogue step): Jeder Dialogschritt umfasst einen Wechselschritt von Aktionen des Menschen und des Computers. Ein Dialogschritt besteht aus Eingabedaten, den zugehörigen Verarbeitungsprozessen und den zugehörigen Ausgabedaten.

Effektivität (effectiveness): Die Genauigkeit und Vollständigkeit, mit der >Benutzer ein bestimmtes Ziel erreichen [ISO-9241-11].

Effizienz (efficiency): Der im Verhältnis zur Genauigkeit und Vollständigkeit eingesetzte Aufwand, mit dem >Benutzer ein bestimmtes Ziel erreichen [ISO 9241-11]. In der Software-Nutzung ist das die Aufgabenangemessenheit, Selbstbeschreibungsfähigkeit, Erwartungskonformität, Lernförderlichkeit, Steuerbarkeit, Fehlertoleranz, Individualisierbarkeit.

Eingabemittel (input device): Als Eingabemittel werden alle peripheren Geräte eines Dialogsystems bezeichnet, die es dem >Benutzer erlauben, dem System Daten zu übermitteln, wie z. B. Tastatur, Tablett, Stift, Maus, Mikrophon, etc.

Erfordernis (implied need): Eigenschaft der Arbeit oder Organisation (>Nutzungs-Kontext), die selbstverständlich als Forderung vorausgesetzt wird, und aus der sich spezifische >Anforderungen ableiten lassen.

Erhärtungs-Prüfung (evidence test): Verifizierung der Vermutung, dass ein identifiziertes Nutzungsproblem durch eine >Normabweichung verursacht ist und die Wirkungen des festgestellten >Mangels nicht zumutbar sind.

Einarbeitungs-Problem (initial usage problem): Eine bei der anfänglichen Nutzung des SW-Produktes festgestellte Barriere, die eine effiziente Erledigung der Arbeitsaufgabe unnötig erschwert, aber bei regelmässiger Nutzung beeinträchtigungsfrei umgangen wird.

Fehler (failure): Ein fehlendes Qualitätsmerkmal, das eine >effektive Nutzung des Produkts verhindert oder ein >Mangel, der als Fehler angesehen wird, weil ihm eine definierte >Anforderung zugrunde liegt.

Funktion (function): Eine Funktion ist eine in sich geschlossene Folge von >Operationen zur Ausführung einer Teilaufgabe des >Benutzers.

Funktionalität (functionality): Gesamtheit aller >Funktionen, die mit dem Dialogsystem ausgeführt werden können, bezogen auf die Arbeitsaufgabe.

Gebrauchstauglichkeit (usability within context of use): Das Ausmass, in dem ein Produkt durch bestimmte >Benutzer in einem bestimmten >Nutzungs-Kontext genutzt werden kann, um bestimmte Ziele >effektiv, >effizient und >zufriedenstellend zu erreichen (ISO 9241-11).

Konformität (conformity/ conformance/ compliance): Übereinstimmung eines Produktmerkmals oder einer am Dialogsystem ausführbaren >Arbeitstätigkeit mit einem >Prüfkriterium. Ist das >Prüfkriterium aus einer Norm abgeleitet (z. B. [ISO-9241-10]), so spricht man im Falle von Übereinstimmung von Normkonformität.

Kontext-Szenario (context scenario): Eine episodische Beschreibung der >Arbeitstätigkeiten einer Arbeitsperson (z. B. der künftige >Benutzer), um den Einfluss der Bedingungen des >Nutzungs-Kontextes auf die Ausführung der Tätigkeiten zu analysieren. Diese Erhebung dient der Vorbereitung der Anforderungs-Entwicklung zu Beginn eines Projekts. Es werden in der Sprache des Benutzers die Nutzungs-Bedingungen eines noch zu entwickelnden SWP

beschrieben, um ein klares Verständnis der >Erfordernisse der Arbeit und Organisation zu erarbeiten. Die sachliche Richtigkeit und Vollständigkeit der Beschreibung muss vom Benutzer bestätigt werden (Validierung).

Mangel (defect): Ein Qualitätsmerkmal, das qualitativ oder quantitativ vom >Prüfkriterium abweicht und somit eine >effiziente Nutzung des Produkts behindert. Ein Mangel kann als nicht zumutbar bewertet werden, wenn die Wirkungen des Mangels nicht umgehbar sind und die >Arbeitstätigkeit des Benutzers beeinträchtigen (siehe >Erhärtungsprüfung).

Marktanalyse (market analysis): Eine Form der Marktforschung zur Feststellung des potentiellen Bedarfs für die Nutzung eines Produkts in einem >Nutzungs-Kontext.

Mensch-Computer-Interaktion (Human-Computer Interaktion): Umfasst ausser >softwareergonomischer Aspekten zusätzlich Nicht-Arbeitssituationen wie "information kiosks" (points of information), Informationstechnologie für Heim und Freizeit, Lernsoftware, die zugehörige Grundlagenforschung und die Entwicklung der erforderlichen Techniken.

Non-Konformität (nonconformity): Die Nichterfüllung eines >Prüfkriteriums (vgl. ISO 8402). Nichterfüllung kann Abwesenheit eines geforderten Qualitätsmerkmals (>Fehler) oder >Abweichung von einem geforderten Tätigkeits-Merkmal (>Mangel) bedeuten. Siehe auch Normabweichung.

Normabweichung (nonconformity): Unterschied zwischen einem gegebenen Produktmerkmal oder einer am Dialogsystem ausführbaren >Arbeitstätigkeit und einem aus der Norm (z.B. [ISO-9241-10]) abgeleiteten >Prüfkriterium. Eine verifizierte Normabweichung wird gemäss dem Prinzip der Falsifikation so lange als vermutete Normabweichung angesehen, bis die >Erhärtungsprüfung zu der Feststellung führt, dass die Normabweichung nicht zumutbare Wirkungen hat. Siehe auch Abweichung, Non-Konformität, Mangel, Fehler.

Nutzungs-Kontext (context of use): Faktoren, die die >Gebrauchstauglichkeit eines Produktes beeinflussen können, z.B. die Qualifikation des >Benutzers, die Merkmale des Arbeitsinhalts, der Arbeitsplanung, der Arbeitsteilung sowie die physische und soziale Umgebung (vgl. [ISO -241-11]).

Nutzungskonzept (useware): Zweck der Nutzung eines SWP und die Gestaltung der Nutzungsweise sowie Arbeits-Aktivitäten. D. h. Die Struktur oder Strategie eines Lösungswegs zur Erledigung einer Aufgabe und die dafür benötigten Hilfsmittel (nicht gemeint als DER eine, ideale Weg der Aufgabenerledigung). Hierbei sind die mentalen Modelle der Nutzer (Benutzer) von der Aufgabe und dem gewünschten Arbeitsergebnis massgebend.

Nutzungsproblem (usage problem): Eine bei der Nutzung des Softwareprodukts festgestellte Barriere, die eine effiziente Erledigung der Arbeitsaufgabe unnötig erschwert, und nicht auf ein >Einarbeitungs-Problem rückführbar ist.

Pflege (usability maintenance): Die Verbesserung eines Softwareprodukts nach Installation am Benutzerarbeitsplatz, um ergonomische Mängel (Abweichungen von [ISO-9241]) zu beseitigen. Die Pflege kann sehr aufwendig sein, wenn versäumt wurde, die >Nutzungsanforderungen im Rahmen der Anforderungs-Entwicklung und des >Prototyping zu validieren.

Produktprüfung (product evaluation): Vergleich der Merkmale eines Produkts (z.B. der >Benutzungsschnittstelle) mit den >Nutzungsanforderungen, um die Erfüllung der Forderungen festzustellen.

Projekt (project): Ein Entwicklungsvorhaben in einer Herstellerorganisation zum Zwecke der Planung, des Entwurfs, der Entwicklung, Dokumentation und >Pflege eines Softwareprodukts.

Prototyping (prototyping): Ein iterativer Prozess zur Herstellung von Teilen einer >Benutzungsschnittstelle anhand eines >Use-Szenarios sowie der Herstellung oder Simulation einer dahinter liegenden Funktionalität zum Zwecke der Veranschaulichung der vom >Designer verstandenen >Nutzungsanforderungen. Dient ein >Prototyp der Erkundung noch nicht verstandener Nutzungsanforderungen, so nennt man diesen Prozess exploratives Prototyping (Usability-Prototyping).

Prüfkriterium (test criterion): Vorzugsweise eine geforderte >Arbeitstätigkeit, die unter Anwendung von [ISO-9241-10] aus einer >Anforderung abgeleitet wurde, oder ein Produktmerkmal, das zu einer geforderten >Arbeitstätigkeit in Bezug steht.

Qualitätssicherung (quality assurance): Alle geplanten, systematischen Massnahmen, die erforderlich sind, um an einem Produkt die geforderten >Prüfkriterien zu erfüllen.

Software-Engineering: Software-Entwicklung als Ingenieurdisziplin. Eine systematische und geplante Vorgehensweise bei der Software-Entwicklung (als Tätigkeit bei der Software-Entwicklung). Systematisierung bei der SW-Erstellung.

Software-Ergonomie: Bezieht sich auf aufgabengesteuerte, zielorientierte Arbeitssituationen. Anwendungsbereiche: Büro/Verwaltung, Prozesssteuerung (Leitstände), Konstruktiosnbüro.

Styleguide: Eine Herstellernorm zur Gestaltung von Oberflächenmerkmalen der >Benutzungsschnittstelle zum Zwecke der konsistenten Gestaltung dieser Merkmale über alle Anwendungsprogramme hinweg

Szenario (scenario): Eine episodische Beschreibung von >Arbeitstätigkeiten unter Berücksichtigung des >Nutzungs-Kontexts (>Kontext-Szenario) oder unter Berücksichtigung der tatsächlichen Ausführung der >Arbeitstätigkeit am Dialogsystem (>Use-Szenario).

Usability-Analytiker: Eine Rolle, die in der Analytischen Usability (UA) tätig ist. Sie bereitet die Gestaltung der >Benutzungsschnittstelle vor, z. B. durch Entwicklung der >Anforderungen mittels >Kontext-Szenarien.

Usability-Engineer: Eine Rolle im >Usability-Engineering, dessen Tätigkeit sich auf die >Anforderungsentwicklung (im >Prototyping) bezieht. Sie setzt die Anforderungen um, die sich aus dem >Nutzungskontext, sowie der Anforderungen an die interaktive Erledigung von Aufgaben in Merkmale des Produkts ergeben.

Usability-Engineering (usability engineering): Eine Entwicklung von >Nutzungsanforderungen, >Prototypen und Softwareprodukten, deren Validierung und Verbesserung, unter systematischer Anwendung von >Usability-Engineering-Methoden im Rahmen eines >Design-Use-Cycle erfolgt.

Usability-Engineering-Methoden (usability-engineering methods): Methoden zur Entwicklung von innovativen Nutzungskonzepten, >Nutzungsanforderungen und >Prototypen, deren Validierung, Bewertung und schrittweise Verbesserung während des Entwurfs- und der >Pflegeprozesses.

Use Case: Die Kurzform eines >Use-Szenarios. Jede Interaktion des >Benutzers mit dem Dialogsystem (>Dialogschritt) wird in zwei Spalten festgehalten, links die >Arbeitstätigkeit des Benutzers, rechts die darauf folgende Reaktion des Dialogsystems.

Use-Szenario (usescenario): Eine episodische Beschreibung von >Arbeitstätigkeiten, die ein >Benutzer zur Erledigung einer Kern-Aufgabe am Dialogsystem durchführt. Diese Erhebung dient der Anforderungs-Entwicklung während des Prototyping oder der Ermittlung von Anforderungen am fertigen Produkt zum Zwecke der Softwareprüfung. Während der Softwareprüfung an einem existierenden System braucht die Interaktion mit dem Dialogsystem nicht

ausführlich erfasst zu werden; es reicht eine Erfassung in Form eines >Use Case aus. >Dialogschritte, die jedoch auf Nutzungsprobleme oder auf vermutete >Normabweichungen hinweisen (critical incidents), sind ausführlich zu beschreiben und anschliessend auszuwerten (Prüfung der >Konformität).

Zufriedenstellung (user satisfaction): Beeinträchtigungsfreiheit der Nutzung und positive Einstellungen der >Benutzer zur Produkt-Nutzung [ISO-9241-11].

8 Literaturverzeichnis

[Acho et.al.´98] C.B.Achour, C.Cauvet, E.Dubois, P.Haumer, P.Heymans, M.Jarke, N.A.M.Maiden, K.Pohl, J.Ralyte, C.Rolland, A.Sutcliffe:
A proposal for a scenario classification framework, Requirements Engineering Journal, 3, No.1, 1998.

[Alba'89] R.und H.Albach:
Das Unternehmen als Institution: Rechtlicher und Gesellschaftlicher Rahmen, Wiesbaden, Gabler Verlag 1989.

[Altm et.al.´93] W.Altmann, E.Doberkat, H.Züllighoven:
Requirements Engineering: Prototyping, German Chapter of the ACM-Band 41, Stuttgart, Teubner Verlag 1993.

[Balz`95] H.Balzert:
Was nützen der Software-Technik die DIN-Normen? In: GI-Softwaretechnik-Trends, Fachgruppen Software-Engineering und Requirements-Engineering, Band 15, Heft 1, Juni 1995.

[Bias/Mayh'94] R.G.Bias, D.J.Mayhew:
Cost-Justifying Usability, Boston, Academic Press 1994.

[Beim et.al.'93] J.Beimel, J.Hüttner, H.Wandke:
Kenntnisse von Programmierern auf dem Gebiet der Software-Ergonomie, Gebert und Winterfeld 1993.

[Berr/Lawr] D.M.Berry, B.Lawrence:
Requirements engineering, IEEE Software, pp. 26-29, Mar./Apr., 1998.

[Bisch/Pomb] W.Bischofsberger, G.Pomberger:
Prototyping-Oriented Software Development. Concepts and Tools, Berlin, Springer Verlag 1992.

[Breu´98] W.Breuer:
Finanzierungstheorie, Wiesbaden, Gabler Verlag 1998.

[Breu´00] W.Breuer:
Investition, Wiesbaden, Gabler Verlag 2000.

[Broo´95] F.P.Brooks Jr.:
Essence and Accident in Software Engineering.The Mythical Man-Month: Essays on Software Engineering (second ed.), Reading, Massachusetts: Addison-Wesley, pp. 179-203, 1995.

[Budd et.al.'92] R.Budde, K.Kautz, K.Kuhlenkamp, H.Züllighoven:
Prototyping – An Aproach to Evolutionary System Development, Exploratory Evolutionary Horizontal and Vertical Prototyping, Berlin, Springer Verlag 1992.

[Carr´95] J.M.Carroll:
Scenario-Based Design: Envisioning Work and Technology in System Development, New York: John Wiley, 1995.

[Carr´94] J.M.Carroll:
Making *use* a design representation, Communications of the ACM, 37/12,pp. 29-35, 1994.

[Carr/Ross´92] J.M.Carroll, M.B.Rosson:
Getting Around the Task-Artifact Cycle: How to Make Claims and Design by Scenario, ACM Trans of Information Systems, vol.10, pp.181-212, 1992.

[Chris/Curti'93] M.B.Chrissis, B.Curtis, C.M.Paulk, C.V.Weber:
Capability Maturity Model for Software, Version 1.1. Technical Report CMU / SEI – 93 – TR – 024, Pittsburgh, February, 1993.

[Chris/Curti'95] M.B.Chrissis, B.Curtis, C.M.Paulk, C.V.Weber:
Capability Maturity Model: Guidelines for Improving the Software Process, Reading, Mass., 1995.

[Coop´99] A. Cooper:
The Inmates are running the asylum , p.15 , Indianapolis, Ind. : Sams, 1999.

[Dahl/Math'95] B.Dahlbom, L.Mathiassen:
Computers in Context: the Philosophy and Practice of Systems Design, Oxford, NCC Blackwell 1995.

[DAT-UE'00] AG Usability Engineering:
DATech Prüfbaustein Usability-Engineering-Prozessqualität, Deutsche Akkreditierungsstelle für Informations- und Telekommunikationstechnik in der TGA (Frankfurt/Main), 2000.

[DAT-G'00] AG Usability Engineering:
DATech Prüfhandbuch Gebrauchstauglichkeit (Version 2.0), Deutsche Akkreditierungsstelle für Informations- und Telekommunikationstechnik in der TGA (Frankfurt/Main), 2000.

[DIN EN 45014] DIN:
Allgemeine Kriterien für Konformitätserklärungen von Anbietern, Berlin, Beuth Verlag 1990.

[Drös'00] W.Dröschel, M.Wiemers:
Das V-Modell 97 - Der Standard für die Entwicklung von IT-Systemen mit Anleitung für den Praxiseinsatz, München, Oldenbourg Verlag 2000.

[Dörn'79] D.Dörner:
Problemlösen als Informationsverarbeitung, Kap. 1.2, Kap. 4,Kap. 5 , Berlin, Kohlhammer Verlag 1979.

[Dzid'00] W.Dzida:
Erkenntnisse, die aufgrund von Gesprächen mit Herrn Dr.W.Dzida entstanden sind, 2000.

[Dzid'99] W.Dzida:
Software-Ergonomie: Softwarenutzung in der betrieblichen Praxis verbessern", IMC`99.

[Dzid'97] W.Dzida:
International User-Interface Standardisation, Computer Science and Engineering Nr.67, CRC Press ACM 1997.

[Dzid/Frei'00] W.Dzida, R.Freitag:
Software-Ergonomie Seminar, Bad Honnef, Deutsche Informatik Akademie 2000.

[Dzid/Frei'98] W.Dzida, R.Freitag:
Making Use of Scenarios for Validating Analysis and Design, IEEE Transactions on Software Engineering, Vol.24, Number 12, S. 1182-1196, Washington 1998.

[Dzid et.al '93] W.Dzida, D.Ackermann, P.Gorny, S.Maass, H.Oberquelle, K-H.Rödiger, W.Rupita, N.Streiz:
SW-Ergonomie-Ausbildung in Informatik-Studiengängen bundesdeutscher Unis, 16, 1, 25-38, Infomatik-Spektrum 1993.

[Dzid et.al '92] W.Dzida, C.Floyd, H.Züllighoven, R.Budde, R. Keil-Slawik:
On controllability In: Software Development and Reality Construction, pp. 325-335, Berlin, Springer 1992.

[Flor/Wino '86] F.Flores, T.Winograd:
Understanding Computers and Cognition - A New Foundation for Design, Norwood, N.J.: Ablex, 1986.

[Floy et.al´92] C.Floyd, H.Züllighoven, R.Budde, R. Keil-Slawik:
Software Development and Reality Construction, Berlin: Springer, 1992.

[Fort/Norm´92] G.Forte, R.J.Norman:
A Self-Assessment by the SE Community, Comm. of the ACM, Vol. 35, No.4, pp.28-36, 1992.

[Gann´97] J.D.Gannon:
Verification and validation, In: A.B. Tucker (ed.): The Computer Science and Engineering Handbook, Boca Raton: CRC Press, chapter 109, 1997.

[Ghez'91] J.Ghezzi, Mandrioli:
"Fundamentals of Software Engineering", Englewood Cliffs, NJ : Prentice Hall 1991.

[Gorny et.al'93] P.Gorny et.al:
Berichte des German Chapter of the ACM, Band 40, Stuttgart, Teubner 1993.

[Gogu´96] J.Goguen:
Requirements Engineering Reconcilation of Technical and Social Issues, technical reports, Oxford University Computing Laboratory, Cambridge UK, 1996.

[Gogu´92] J.Goguen:
Requirements Engineering Reconciliation of Technical and Social Issues, tech. report- Centre for Requirements and Foundations, Oxford University Computing Lab, Cambridge, U.K., 1992.

[Gryc et al.´93] G.Gryczan, K.Kilberth, H.Züllighoven:
Objektorientierte Anwendungsentwicklung: Konzepte, Strategien, Erfahrungen, Kap. 5, Wiesbaden, Vieweg Verlag 1993.

[W.Hacker´87] W. Hacker:
Software-Ergonomie: Gestalten rechnergestützter geistiger Arbeit?!, In: W.Schönpflug und

M. Wittstock (Hreg.): SW-Ergonomie´87. german Chepter of the ACM, bericht Nr. 29., S 31-54, B. G. Teubner, Stuttgart 1987.

[Hsia et.al ´93] P.Hsia, A.Davis, D.Kung, :
Status Report: Requirements Engineering, IEEE Software, pp. 75-79, 1993.

[Hill/Houd´97] C.Hill, S.Houde:
What do Prototypes Prototyping, In: M.Helande et al. , Handbook of HCI, S. 367-381, Elsevier Amsterdam 1997.

[Isen et.al '96] S.Isensee, J.Rudd, K.Stern:
Low High Fidelity Prototyping Derbate in: ACM Interactions, S.76-85, 1997.

[ISO-12207] International Organization for Standardization:
Software life cycle processes, Berlin, Beuth Verlag 1996.

[ISO-12119] International Organization for Standardization:
SW-Erzeugnisse: Qualitätsanfordereungen und Prüfbestimmungen (früher DIN 66285), Berlin, Beuth Verlag 1995.

[ISO-13407] International Organization for Standardization:
Human centered design processes for interactive systems, Berlin, Beuth Verlag 1999.

[ISO-15504-2] International Organization for Standardization:
Information Technologie - SW Process Assessment: reference model for processes and process capability, Berlin, Beuth Verlag 1998.

[ISO-15504-5] International Organization for Standardization:
Information Technologie - SW Process Assessment: an assessment model and indicator guidance, Berlin, Beuth Verlag 1998.

[ISO-9241-x] International Organization for Standardization:
Qualitätsmanagement-Systeme, Berlin, Beuth Verlag 1994.

[ISO-9241-x] International Organization for Standardization:
Ergonomic requirements for office work with visual display terminals; Parts 1-17, Berlin, Beuth Verlag 1992-1996.

[ISO-9126] International Organization for Standardization:
Software product evaluation - Quality characteristics and guidelines for their use (Beurteilung von SWP, Qualitätsmerkmale und Leitfaden zu deren Verwendung), Berlin, Beuth Verlag 1991.

[ISO-8402] International Organization for Standardization:
Qualitätsmanagement Begriffe, Berlin, Beuth Verlag 1999.

[ISO-DIN- 45020] International Organization for Standardization:
Allgemeine Fachausdrücke unde deren Definitionen betreffend Normung und damit zusammenhängenede Tätigkeiten, Berlin, Beuth Verlag 1999.

[Jaco et.al'99] I.Jacobson et.al:
The Unified Software Development Process, New York, Addison-Wesley 1999.

[Jark et.al'98] M.Jarke, P.Haumer, K.Pohl, K.Weidenhaupt:
Scenarios in System Development: Current practice, IEEE Software, pp. 34-45, March/April, 1998.

[Kang/Wood´92] K.Kang, D.Wood:
Classification and Bibliography of Software Prototyping, Technical Report, CM/SEI-92-TR-13, 1997.

[KBSt´97] Koordinierung- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung:
Entwicklungsstandard für IT-Systeme des Bundes (V-Modell), Teil 1, 2, 3, Bundesministerium des Innen, Verlag Bundesanzeiger PF 100534, 50445Köln 1997.

[Kyng95] M.Kyng:
Scenario-based Design: Envisioning Work and Technology in System Development - Creating Context for Desing, Chapter 4, John M. Carroll, New York, J.Wiley & Sons 1995.

[Kyng/Math´97] M.Kyng, L.Mathiassen:
Computers and Design in Context, The MIT Press, Cambridge, MA, 1997.

[Land'95] T.K.Landauer:
The Trouble with Computers: Usefulness, Usability, and Productivity, The MIT Press, Published Cambridge 1995.

[Laur´90] B.Laurel:
The Art of HCI design, Reading, MA, Addison-Wesley 1990.

[Louc/Kara´95] P.Loucopoulos, V.Karakostas:
System Requirements Engineering, 1st ed.. London: McGraw Hill, 1995.

[Mack´95] R.L.Mack:
Discussion: Scenarios as Engines of Design, In: J.M. Carroll (ed.): Scenario-Based Design. Envisioning Work and Technology in System Development, New York: John Wiley, pp. 361-386, 1995.

[Manh´98] M.Manhartsberger:
Prototyping – Theorie und Praxis, Ergonomie&Informatik 1998.

[Mina'00] M.Minasi:
The software conspiracy. Why software companies put out faulty products, how they can hurt you, and what you can do about it, New York, McGraw-Hill 2000.

[Müll/West'94] C.D.Müller-Hengstenberg, F.Graf von Westphalen:
DV-Projektrecht, S-56, Berlin, Springer Verlag 1994.

[Niel'93] J.Nielsen:
Usability Engineering, Boston, AP Professional 1993.

[Norm´98] D.Norman:
The invisible Computer; why good products can fail, the pc is so comlex, and information appliances are the solution, The MIT Press, Cambridge, MA, 1998.

[Norm/Drap´86] D.Norman, S.Draper:
User centred system design: new perspectiv on HCI, Hillsdale, NJ: Erlbaum Lawrence 1986.

[Ober´99] H.Oberquelle:
Useware Design and Evolution: Bridging Social Thinking and SW Construction, manuskript, Hamburg 1999.

[Pree et al.´94] J.Preece et al.:
HCI, Kapitel 27, New York, Addison-Wesley 1994.

[Sabe'97] H.Sabel, Ch.Weiser:
Dynamiken im Marketing: Umfeld-Strategie-Struktur-Kultur, Wiesbaden, Gabler Verlag 1997.

[Sidd´94] J.Siddiqi:
Challenging universal truth of requirements engineering, IEEE Software, pp.18-19, 1994.

[Szwi´97] G.Szwillus:
Prototypen für Benutzungsschnittstellen, Tagungsband Padeborn PB (Hrsg. Szwillus) November, 1997.

[Thal'00] G.E.Thaller:
ISO-9001 Softwareentwicklung in der Praxis, Hannover, Heise Verlag 2000.

[Thal´90] G.E.Thaller:
Software-Qualität: Entwicklung, Test, Sicherung, Düsseldorf, Sybex 1990.

[VDI-2221] Richtlinien:
Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte, Düsseldorf, VDI Verlag 1985.

[Vers'00] G.Versteegen:
Das V-Modell in der Praxis, Heidelberg, dpunkt Verlag 2000.

[Weis/Brow´97]M.Weiser, J.S.Brown:
The Coming Age of Calm Technology, In: Denning, P.J., Metcalfe, R.M. (eds.). Beyond Calculation. The Next Fifty Years of Computing. Copernicus/ Springer, New York, 75-85, 1997.

[Wino et.al'96] T.Winograd, J.Bennet, De Young, B.Hartfield:
Bringing Design to Software, ACM Press, New York / Addison-Wesley, Reading, MA, 1996.

[Wino'97] T.Winograd:
The Design of Interaction, In: Denning, P.J., Metcalfe, R.M. (eds.). Beyond Calculation. The Next Fifty Years of Computing, Springer, New York, 149-161, 1997.

[Winz'00] P.Winzer:
Die neue ISO-9000:2000, Interpretation der Forderungen des Normentwurfes E-DIN-EN-ISO-9000: 1999-05, Köln, Fachbroschüre TÜV 2000.

[Zühl/Wahl´99] D.Zühlke, M.Wahl:
Hardware, Software – Useware. Maschinenbedienung in vernetzten Produktionssystemen, Elektronik 23, 54-62, 1999.

[Züll'98] H.Züllinghoven:
Das Objektorientierte Konstruktionshandbuch:WAM, Heidelberg, dpunkt Verlag 1998.

[URL:bma]:
Verordnung über Sicherheit und Gesundheitsschutz bei der Arbeit an Bildschirmgeräten,
http://www.bma.de/download/gesetze/Bildschirmarbeit.htm

[URL:12207]:
ISO-12207, http://www.informatik.uni-bremen.de

[URL:Niel]:
Jakob Nielsen Website, http://www.useit.com/

[URL:Disc] Jakob Nielsen :
The Discount Usability Engineering Approach ,
http://www.useit.com/papers/guerrilla_hci.html

[URL:Usab]:
Usability, http://atwww.hhi.de/draft/USINACTSnorm/whatis.html

[URL:V-Model]:
Das Vorgehensmodell, http://www.scope.gmd.de/vmodel/

[URL:ZENO]:
GMD-ZENO, http://zeno/zeno/open/