Universität Hamburg
Fachbereich Informatik

Integration ausgewählter softwaretechnischer Ansätze in das MICROPOLIS-Modell

Diplomarbeit von Natalie Daou

Erstbetreuer: Prof. Arno Rolf (ASI)
Zweitbetreuer: Prof. Christiane Floyd (SWT)

Februar 2005
Hamburg

Diplomarbeit am
Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Straße 30
22527 Hamburg

Vorgelegt von
Natalie Daou
Matrikelnr.: 4816012

Betreut durch
Prof. Dr. Arno Rolf (Erstbetreuer)
Universität Hamburg
Fachbereich Informatik
Arbeitsbereich Angewandte- und Sozialorientierte Informatik

Prof. Dr. Christiane Floyd (Zweitbetreuerin)
Universität Hamburg
Fachbereich Informatik
Arbeitsbereich Softwaretechnik

Danksagung

Ich möchte mich an dieser Stelle bei allen Personen bedanken, die an dieser Diplomarbeit direkt durch Diskussionen über das Thema, Korrekturlesen oder auch indirekt durch Ihre Unterstützung beteiligt waren.

Meinen Dank möchte ich folgenden Personen aussprechen:

- Meinem Erstbetreuer Herrn Professor *Arno Rolf*, dem ich das Thema meiner Diplomarbeit verdanke und dessen Modell die Grundlage dieser Arbeit liefert.
- Meiner Zweitbetreuerin Frau Professor *Christiane Floyd*, die durch die arbeitsbereichsübergreifende Zusammenarbeit diese Arbeit ermöglichte.
- *Dorina Gumm*, die mir stets als kompetente Ansprechpartnerin zur Verfügung stand.
- *Katrin Brümmer*, die meine Arbeit auf sprachliche Feinheiten Korrektur las.

Persönlich möchte ich mich, über die Diplomarbeit hinaus, noch bei *Familie Eggers* und *Familie Ernst* bedanken, die mir in den vergangenen Jahren zur Seite standen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Das Thema dieser Diplomarbeit umfasst Inhalte aus zwei Arbeitsbereichen der Informatik an der Universität Hamburg. Zum einen aus dem Arbeitsbereich „Angewandte und Sozialorientierte Informatik“ (ASI), zum anderen Ansätze aus dem Arbeitsbereich „Softwaretechnik“ (SWT). Das MICROPOLIS-Modell, entwickelt am Arbeitsbereich ASI, hilft der Informatik bei der Einordnung in den gesamtgesellschaftlichen Kontext.

Der Titel der Diplomarbeit „Integration ausgewählter softwaretechnischer Ansätze in das MICROPOLIS-Modell“ beinhaltet eine Verknüpfung der Softwaretechnik mit den Erkenntnissen eines gesamtgesellschaftlichen Kontextes.

Das Ergebnis meiner Betrachtungen des MICROPOLIS-Modells und den softwaretechnischen Ansätzen ist die vorliegende Diplomarbeit.

1.1 Kontext und Motivation

Meine Motivation, meine Arbeit diesem Thema zu widmen, resultiert aus meinem Ergänzungsfach Betriebswirtschaftslehre und meinem Vertiefungsgebiet Softwaretechnik. Beide Fächer verbinde ich in meiner Arbeit miteinander, da sie sich sinnvoll ergänzen. In der Betriebswirtschaftslehre habe ich den Schwerpunkt „Betriebswirtschaftliche Datenverarbeitung“ (BDV) gewählt, der sich unter anderem mit Computergestützten Informations- und Planungssystemen, sowie dem Entwurf und der Planung von Softwaresystemen befasst. In diesem Kontext bin ich auf Fragen gestoßen, die ich näher erörtern möchte:

- Wie können gesellschaftliche Zusammenhänge mit betriebswirtschaftlichen Belangen, genauer gesagt organisationale Bedürfnisse, durch Informatiksysteme vereint werden?
- Wie ordnet sich die Informatik in die Gesellschaft ein?

Das MICROPLIS-Modell trägt zur Lösung der beider Problemstellung bei. Es geht aber einen Schritt weiter und bezieht die Informatik in diese Fragestellung mit ein. Eine

genauere Erläuterung des MICROPOLIS-Modells findet sich in Kapitel 3. Die betriebswirtschaftlichen Belange werden durch die Betrachtung von Organisationen gegenübergestellt zu der Entwicklung von Informatiksysteme implizit berücksichtigt.

In Organisationen werden Entscheidungen über Informatiksysteme sowohl aus technischen Gründen als auch aus strategischen Gründen gefällt. Wenn man sich aus betriebswirtschaftlicher Sicht mit Informatiksystemen in Organisationen beschäftigt, dann ist es sinnvoll, die Informatiksysteme selbst näher zu betrachten. Die Begegnung zwischen Informatik und der Gesellschaft ist ausgesprochen interessant, da es die Auswirkungen der Informatik auf Gesellschaft betont und die Konsequenzen der Handlungen von Informatikern aufzeigt. Das MICROPOLIS-Modell ist somit ein Bindeglied der Disziplinen Wirtschaft und Informatik.

Die Wirtschaftsinformatik befasst sich mit der Konzeption, Entwicklung, Einführung, Wartung und Nutzung von computergestützten Informationssystemen in Organisationen. Sie geht über die Schnittmenge aus BWL, Informatik und Technik hinaus.
Die Informatik ist eine junge Wissenschaft, die verschiedene Ausprägungen und Fachbereiche entwickelt hat. Die Fachbereiche sind Teilgebiete innerhalb der Informatik, die sich in vier Teilbereiche gliedert. Diese sind die Theoretische Informatik, die Angewandte Informatik, die Technische Informatik und die Praktische Informatik. Die Softwaretechnik ist eine Vertiefung aus dem Teilbereich der Praktischen Informatik [Rolf 1998].

Hieraus ergeben sich die Aufgaben, die Informatiker erfüllen müssen. Ferner sollten sie sich auch die Konsequenzen ihrer Arbeit verdeutlichen wie auch das folgende Zitat zeigt.
Nach [Rolf und Siefkes 1992] manipulieren Informatiker ökonomische und soziale Systeme, indem sie Rechner in solche Systeme hineinkonstruieren. Die Autoren werfen dabei die Frage auf, wie Wissenschaft und Technik weiter zu bringen sind, unter Einbeziehung sozialer Umfelder. Rolf geht einen Schritt weiter und plädiert dafür die Informatik als *Gestaltungswissenschaft“* anzusehen [Rolf und Siefkes 1992].

Um diesen Anspruch an die Informatik zu erfüllen, entwickelt Rolf den Orientierungsrahmen MICROPOLIS-Modell für die Organisations- und Wirtschaftsinformatik, den er ursprünglich OWI-Gestaltungsmodell nannte (siehe [Rolf 1998], S.147 – 168).

In Bezug darauf stellt er in [Rolf 1998] die folgenden Thesen auf:

These 1

> *„Bei der „Modellierung von Informationssystemen“ greift die Informationstechnik in organisatorische und soziale Phänomene ein und verändert sie. Dabei müssen soziale und technische Entwicklungspotentiale austariert werden.“*
> (siehe [Rolf 1998], S. 147)

These 2

> *„Gestaltungsansätze und Softwareprodukte in Organisationen unterscheiden sich aufgrund der gewählten Perspektiven und der zugrundeliegenden Organisationsleitbilder. Die Verknüpfung der Perspektiven wird bislang als unproblematisch angesehen, da die sog. „Ebenen“ in einem hierarchischen Verhältnis zueinander stehen (Top-Down-Sicht) oder scheinbar ohne weiteres „von unten nach oben“ zusammengefügt werden können (Bottom-up-Sicht).“*
> (siehe [Rolf 1998], S. 148)

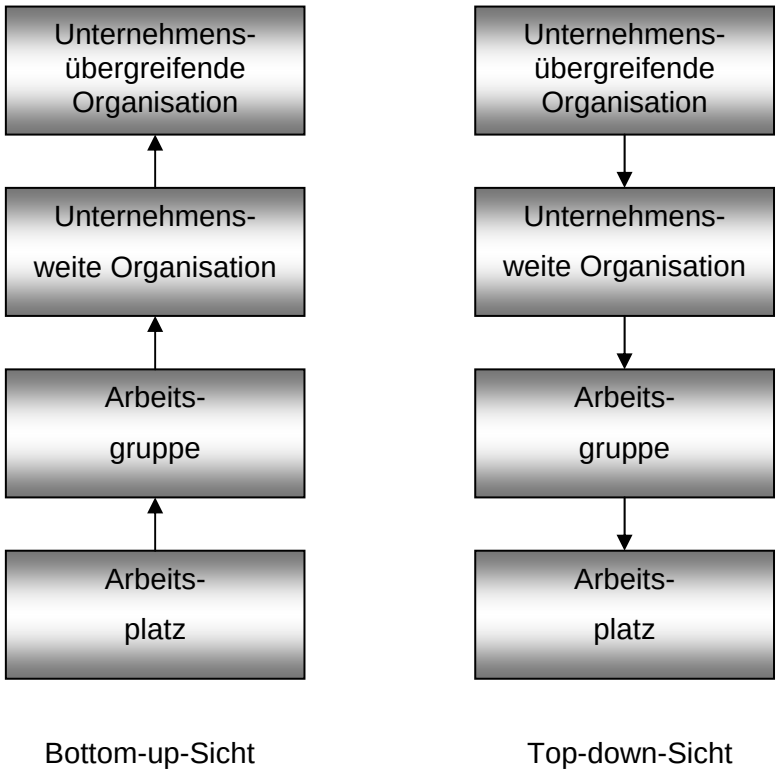

Abbildung 1-1: OWI-Gestalltungmodell: Bottom-up-Sicht und Top-down-Sicht Nach: [Rolf 1998]

In der Abbildung 1-1 werden unterschiedlichen Sichtweisen Bottom-up und Top-down dargestellt. Mit Top-down-Sicht wird die betriebliche Organisation in ihrer Gesamtheit

modelliert und gesteuert. Management-Informationssysteme (MIS) haben als erstes versucht diese umzusetzen, scheiterten aber an der Komplexität. Ein weiteres Problem war auch, dass die Benutzer nicht ausreichend in die Entwicklung einbezogen wurden.
Die Bottom-up-Sicht hingegen wird vom Individuum bzw. Einzelarbeitsplatz ausgegangen. Bei dieser Sicht werden die Wechselwirkungen zwischen Individuum und Technik fokussiert. Für das einzelne Individuum ist die Organisation als Ganzes nur schwer nachvollziehbar. Eine IT-unterstützende Arbeitsgruppe steht zwischen diesen beiden Sichten. Hier werden sowohl die Gruppen als auch den Zusammenhang zum Gesamten betrachtet [Rolf 1998].

1.2 Problembeschreibung

In seinem Modell wirft Rolf folgende Frage auf:

> „*Wie könnte ein analytischer Orientierungsrahmen aussehen, der die vorherrschende Perspektive des hochspezialisierten Expertenwissens so ergänzt, dass die Akteure der Informatik in Wissenschaft und Praxis in ihren Kommunikations- und Kooperationsbeziehungen und in ihrem Weltverständnis und ihrer Weltgestaltung unterstützt werden?*“ (siehe [Rolf 2003], S.3)

Der von ihm entwickelte Orientierungsrahmen, namens MICROPOLIS-Modell, verbindet die Informatik mit Anwendungen in Organisationen mit gesellschaftlichen Kontexten und setzt diese in einen Zusammenhang zueinander. Dieser Orientierungsrahmen beantwortet die von Rolf gestellte Frage.
Das MICROPOLIS-Modell betrachtet die Informatik aus drei Perspektiven, beginnend mit der Mikro-Perspektive, die den Kern der Informatik betrachtet. Darauf folgend bezieht die Meso-Perspektive die Umgebung der Informatik, d.h. Organisationen und Akteure mit ein. Die letzte Perspektive, die Makro-Perspektive zieht zusätzlich die Globale Gesellschaft in den Fokus. Eine nähere Beschreibung folgt in Kapitel 3.
Im Prozess der Softwareentwicklung bezieht das MICROPOLIS-Modell die handelnden Akteure in die Perspektiven mit ein. Diese befinden sich in sogenannten Arenen bzw. Handlungsfeldern, in denen es zu Kooperationen und Konflikten kommen kann [Rolf 2004a+b].
Die Softwaretechnik bringt verschiedene softwaretechnische Ansätze zur Softwareentwicklung hervor. Diese haben in der Umsetzung ganz unterschiedliche Gestaltungsformen.

Dem MICROPOLIS-Modell fehlen diese konstruktiven Gestaltungsformen, sowie Ansätze, Modelle und Methoden der Softwaretechnik bzw. Wirtschaftsinformatik. An diesem Punkt setzt nun diese Diplomarbeit an.

1.3 Zielsetzung

Ziel dieser Diplomarbeit ist es, einige ausgewählte softwaretechnische Ansätze im MICROPOLIS-Modell zu diskutieren, um damit exemplarisch die Möglichkeiten des MICROPOLIS-Modells darzustellen.

Hierzu sollen ausgewählte softwaretechnische Ansätze untersucht und mit dem MICROPOLIS-Modell so verknüpft werden, dass Orientierungs- und Gestaltungsperspektiven einander ergänzen.
Softwaretechnische Ansätze sind Methoden, Techniken und Modelle und dienen dem Entwurf, der Planung, der Realisierung und der Implementierung von Softwareprodukten. Dieses Wissen unterstützt und verbessert das Verständnis für die Beurteilung von Informatiksystemen.

Die Arbeitsbereiche Angewandte- und Sozialorientierte Informatik und Softwaretechnik sollen fachlich durch das MICROPOLIS-Modell so verbunden werden, dass Informationen über das konkrete fachliche Wissen, dem Verfügungswissen (siehe Kapitel 2.3 Was ist ein Orientierungsrahmen?) hinaus transportiert werden. Die Ansätze der Softwaretechnik sollen einerseits dem MICROPOLIS-Modell Gestaltungsmöglichkeiten bieten. Andererseits soll das MICROPOLIS-Modell der Softwaretechnik einen übergeordneten Kontext bieten, der zu dem spezialisierten Verfügungswissen auch Orientierungswissen (siehe Kapitel 2.3) vermittelt. Diese Schnittstelle erörtere ich in der vorliegenden Arbeit.

Die Aufgabe dieser Diplomarbeit ist es also, das MICROPOLIS-Modell durch die Anbindung einer Auswahl etablierter softwaretechnischer Ansätze zu ergänzen, um so exemplarisch die Integration von Verfügungs- und Orientierungswissen in der Informatik zu demonstrieren.
Der Fokus wird auf der mittleren, der Meso-Perspektive (siehe Kapitel 3.2) liegen, da sich dort die softwaretechnischen Ansätze einordnen lassen und ihre Anwendung finden. Die Meso-Perspektive befasst sich mit Informatiksystemen in Organisationen.

Die Softwaretechnik befasst sich mit dem Entwurf, der Planung, Entwicklung und Implementierung von Informatiksystemen und verwendet dafür verschiedene Ansätze.

Abschließend ist zur Vorgehensweise zu sagen, dass die Diplomarbeit auf Literaturarbeit basiert.

1.4 Gliederung der Arbeit

Ausgehend von dem MICROPOLIS-Modell und einer Auswahl softwaretechnischer Methoden beschreibt diese Diplomarbeit eine Verbindung der Softwaretechnik mit der Angewandten- und Sozialorientierten Informatik.

Die Aufgabe hierbei ist es, ausgewählte softwaretechnische Methoden zu untersuchen und mit dem MICROPOLIS-Modell zu verknüpfen, so dass mögliche Orientierungs- und Gestaltungsperspektiven aufgezeigt werden.

Hierfür werde ich in **Kapitel 2** Definitionen wichtiger Begriffe für diese Arbeit erläutern.

Im **Kapitel 3** wird das MICROPOLIS-Modell mit seinen drei Ebenen beschrieben. Dazu werde ich jeweils die Relevanz der Perspektiven herausarbeiten.

Bevor ich im **Kapitel 4** zu den „ausgewählten softwaretechnischen Ansätzen“ komme, werde ich Definitionen festlegen, auf denen die Arbeit aufbaut. Dabei werde ich mit der Frage „Was ist Softwaretechnik?“ den von mir betrachteten Bereich der Softwaretechnik aufgreifen. Die Frage nach der Methode und der Technik ist elementar für die Auswahl der Ansätze. Aus diesem Grund erfolgen Erläuterungen und Beschreibungen verschiedener Methoden, Techniken, Modelle. Das Kapitel 4 ist der Begründung und Verdeutlichung verwendeter Ansätze vorbehalten. Ich werde diesen Bereich nutzen, um zu beschreiben aus welchem Grund ich mich für die in diesem Kapitel beschriebenen softwaretechnischen Ansätze entschieden habe.

Die Diskussion der Softwaretechnik im MICROPOLIS-Modell und damit der Kern dieser Diplomarbeit, findet im **Kapitel 5** statt. Dieses Kapitel ist unterteilt in zwei Teile. Im ersten Teil erörtere ich, ob das MICROPOLIS-Modell Hinweise für

softwaretechnische Fragestellungen geben kann. Hierzu werde ich die folgenden beiden Fragen bearbeiten:

1. Kann das MICROPOLIS-Modell Hinweise für softwaretechnische Fragestellungen geben?
2. Wie könnte eine Integration von STEPS und MICROPOLIS-Modell gelingen?

In dem zweiten Teil des Kapitels 5 befasse ich mich mit der Frage, wie exemplarisch eine Integration von dem softwaretechnischen Ansatz STEPS (Softwaretechnik für Evolutionäre Partizipative Softwareentwicklung) und MICROPOLIS-Modell gelingen könnte.

In der Schlussbetrachtung in **Kapitel 6** fasse ich meine Ergebnisse zusammen, gehe auf entstandene Fragen ein und gebe einen Ausblick.

2 Begriffe

Bevor ich das MICROPOLIS-Modell erläutere, ist es wichtig, jeweils eine Definition der Begriffe „Modell“ und „Methode“ festzulegen. Hierfür beziehe ich mich auf bereits bestehende Definitionen und erläutere, welche Ausführungen ich in dieser Diplomarbeit zugrunde lege.

2.1 Was ist ein Modell?

In der Gesellschaft für Informatik e.V. (GI) gibt es Arbeitskreise, die sich mit der Strukturierung und Abgrenzung von Begriffen in der Informatik befassen. Hierzu greifen die Autoren auf bekannte Literatur zurück und ergänzen diese mit eigenen Erläuterungen. Das Ergebnis dieser Arbeitskreise ist ein Begriffsnetz für die Informatik, das im Internet für jeden zugänglich ist. Dies macht deutlich, dass es einen Bedarf genauer Definitionen in der Informatik gibt.

In diesem Begriffsnetz wird ein Modell definiert als eine idealisierte und vereinfachte Nachahmung eines dienenden Vorbildes. Die Darstellung des Modells ähnelt dem zu modellierenden Gegenstand, System oder Weltausschnittes. Ziel des Modells ist es, die Eigenschaften des Vorbildes besser studieren zu können.
Ein Gegenstand ist ein dem Menschen „e*ntgegenstehendes*“ abstraktes oder konkretes Ding. Dieses trägt einen Namen, durch den es eindeutig identifizierbar ist. Eine Beschreibung des Gegenstandes kann diesen mit anderen Gegenständen gleichsetzen oder unterscheiden bzw. klassifizieren [Informatik-Begriffsnetz 2005]. Diese Bedeutung des Begriffs Modell ist bei [Hesse et al. 1994] ebenso zu finden (siehe [Hesse et al. 1994], S. 40 und 42).

Nach [Rolf 1998] dient ein Modell in einem wissenschaftlichen Kontext zur Beschreibung und Untersuchung eines Systems. Die Grundidee bei der Formulierung eines wissenschaftlichen Modells ist die Reduktion von Komplexität. Er schreibt zum Modellbegriff, dass die gesamte Wissenschaft durch ein Denken in Modellen geprägt ist. Ferner unterscheidet er Erklärungs- und Gestaltungsmodelle. Erklärungsmodelle

dienen der Analyse konkreter Systeme und sind somit Interpretationsschemata. Gestaltungsmodelle erfüllen den Sinn der praktischen Umsetzung von theoretischem Wissen (siehe [Rolf 1998], S.18).

Ergänzend zu den bisherigen Definitionen sieht [Schefe 1999] ein Modell sowohl als Nachbild, als auch zugleich als Vorbild. Ein Nachbild des zu modellierenden Gegenstandes, sowie ein Vorbild aus dem Erkenntnisse gewonnen werden können. (siehe [Schefe 1999], S.132)

All diesen Definitionen liegt der Gedanke zugrunde, dass ein Modell eine vereinfachte Nachahmung eines dienenden Vorbildes ist. Modelle reduzieren die Komplexität der Realität und sind somit ein Hilfsmittel in der Wissenschaft. Da ein Modell auch dazu dient, die Eigenschaften des Vorbildes zu erforschen, stimme ich dem Dualitätsgedanken von [Schefe 1999] zu.

2.2 Was ist eine Methode?

Für Floyd und Züllighoven liefern Methoden die Vorgaben für ein systematisches Vorgehen bei der Softwareentwicklung. Der Begriff wird sowohl auch auf die Softwareentwicklung, als auch auf einzelne Aktivitäten wie Analyse, Entwurf oder Programmierung bezogen [Floyd und Züllighoven 1997].

Darüber hinaus betrachtet [Andersen et al. 1990] Methoden in Bezug auf einen Einsatzbereich. In Form von Organisationsformen, Mitteln und Techniken liefern Methoden Richtlinien und werden als eine Sicht der Softwareentwicklung definiert. Eine Methode besteht demnach aus Techniken.

Um die Wichtigkeit des Begriffs „Methode“ zu demonstrieren, greife ich an dieser Stelle eine aktuelle Diskussion von [Floyd und Pape 2004] auf.

In dem Artikel [Floyd und Pape 2004] wird anhand des CommSy-Projektes an der Universität Hamburg werden über die Softwareentwicklung erörtert und diskutiert. Ich bediene mich dieser Diskussion, da sie viel über die Softwareentwicklung in einem Team erklärt und die unterschiedlichen Sichtweisen informativ darstellt. Die Autoren verwenden die Synonyme Methode, Vorgehensmodell und methodische Züge. Die

einzelnen Erläuterungen dieser Begriffe dienen dazu, die Funktionalität der Begriffe für die Softwaretechnik aufzuzeigen.

Bernd Pape sieht eine Methode im Sinne eines Vorgehensmodells. Ein *„regelmäßiges und regelhaftes Vorgehen“* definiert er als *„methodische Züge“ (siehe* [Floyd und Pape 2004], S. 397)

Laut Wolf-Gideon Bleek wird ein Vorgehen erst dann zu einer Methode, wenn es mindestens ein zweites Mal vollzogen wurde. *(siehe* [Floyd und Pape 2004], S. 397)

Welche Bedeutung hat eine Methode in der Softwaretechnik?
Methoden dienen in der Softwaretechnik als Vorschriften für den Entwurf und die Planung, sowie für die Entwicklung von Softwareprojekten. Methoden können geschult, eingeführt und überprüft werden [Floyd und Pape 2004].

Nach Lucy Suchman sind Methoden wie Ressourcen anzusehen, die im Rahmen des Entwurfs und der Programmierung gewählt werden können, so dass die Freiheit besteht zwischen unterschiedlichen Schritten im Vorgehen zu entscheiden [Floyd und Pape 2004].

Christiane Floyd definiert die Methode weiter. Sie sagt, dass man erst wenn ein Vorgehen benannt und beschrieben wird, eine Methode erhält, die in einer lehrbaren Form weitergegeben werden kann [Floyd und Pape 2004].

Lars Mathiassen definiert Methode hingegen durch ein bestimmtes Anwendungsgebiet, eine Perspektive oder als eine empfohlene Richtlinie zur Softwareentwicklung (Darstellungs-Techniken und Organisationsformen) [Floyd und Pape 2004].

In dem oben genannten Artikel werden als Beispiele für Methoden zum einen Nutzungsszenarien und zum anderen der Autor-Kritiker-Zyklus genannt.

Ich werde den Begriff „Methode“ in dieser Arbeit wie folgt verwenden:
Eine Methode ist eine Vorgehensweise, die für den Prozess der Softwareentwicklung eine empfohlene Vorschrift oder Richtlinie bietet, diese zu planen und umzusetzen.
Ich halte diese Definition für adäquat, da sie den Kern der vorrangegangenen Aussagen in sich vereint.

2.3 Was ist ein Orientierungsrahmen?

Wie schon erwähnt, ist das MICROPOLIS-Modell ein Orientierungsrahmen. Was aber ist ein Orientierungsrahmen? Wozu dient ein Orientierungsrahmen? Im folgenden Abschnitt gehe ich auf diese Fragen ein, um den Sinn eines Orientierungsrahmens zu demonstrieren und damit den Nutzen des MICROPOLIS-Modell besser erörtern zu können.
Der [Duden1990] versteht unter dem Begriff „Orientierung“ ein „Zurechtfinden im Raum“ bzw. auch „eine geistige Einstellung oder Ausrichtung“.
Das MICROPOLIS-Modell ermöglicht eine Orientierung der Informatik in der Gesellschaft. Hierzu werden Akteure aus verschiedenen Perspektiven beobachtet. Die Kommunikation dieser Akteure geht über die Grenzen der Perspektiven hinaus, ohne die Art der Mittel festzulegen [Rolf 2004a+b].
Daraus schließe ich, dass ein Orientierungsrahmen die Objekte der Beobachtung in einen Gesamtkontext einordnet, ohne eine Methode oder eine Technik vorzugeben.
Sinn und Zweck eines Orientierungsrahmens ist es demnach, dem Betrachter eines solchen, ein Mittel an die Hand zu geben, die ihm die Orientierung in einem größeren Kontext ermöglicht.

Mittelstrass definiert Orientierungs- und Verfügungswissen wie folgt:

> *„Verfügungswissen ist ein positives Wissen, ein Wissen um Ursachen, Wirkungen und Mittel, Orientierungswissen ist ein regulatives Wissen, ein Wissen um Ziele und Maximen.“ (siehe [Mittelstrass 1989], S. 19)*

Verfügungswissen ist also ein konkretes, konstruktiv fachliches Wissen. In diesen Bereich fallen Methoden und Techniken. Dieses Wissen ist rational und technisch.
Orientierungswissen hingegen versteht man als das Wissen, das eingebettet in den sozialen Kontext, die Ziele und Grundsätze beinhaltet.

Das MICROPOLIS-Modell vereint beides in sich. Es verknüpft das notwendige fachspezifische Verfügungswissen und setzt dieses in einen sozialen Kontext, so dass beides zu Orientierungswissen kumuliert wird.

Orientierungswissen ermöglicht die Einordnung des eigenen fachlichen Handelns.

3 MICROPOLIS-Modell

In diesem Kapitel wird das MICROPOLIS-Modell dargestellt und erläutert. Zunächst wird auf die Entstehung und die damit verbundenen Notwendigkeit dieses Orientierungsrahmens eingegangen. Im Verlauf des Kapitels werden die einzelnen Perspektiven demonstriert.

Ursprünglich bezeichnete Rolf diesen Orientierungsrahmen als OWI-Gestaltungsmodell [Rolf 1998] oder auch als ISO/OWI-Orientierungsrahmen [Rolf 2002]. ISO steht für Informatiksysteme in Organisationen. OWI ist das Kürzel für Organisations-– und Wirtschaftsinformatik.
Um eine Verwechslung durch die Kürzel mit dem ISO-Standard auszuschließen, war es erforderlich diesen Orientierungsrahmen einen neuen Namen zu geben.
In [Rolf 2004 a+b] nennt der Autor es MICROPOLIS-Modell. Wenn ich in dieser Arbeit den Begriff MICROPOLIS-Modell verwende, meine ich den im Folgenden beschriebenen Orientierungsrahmen. Hierzu gebe ich erst mal einen Überblick über die Hintergrund des MICROPOLIS-Modells und die Entstehung der eben genannten Begriffe.

Hintergrund und Entstehung des MICROPOLIS-Modells

Der ISO/OWI – Orientierungsrahmen ist ein Arbeitskonstrukt der anwendungsnahen Informatik für die IT- unterstütze Organisationsgestaltung. Ihm liegt die Organisations- und Wirtschafts-Informatik (OWI) mit der Metapher „Informatiksysteme für Organisationen (ISO)“ zugrunde [Rolf 2002].

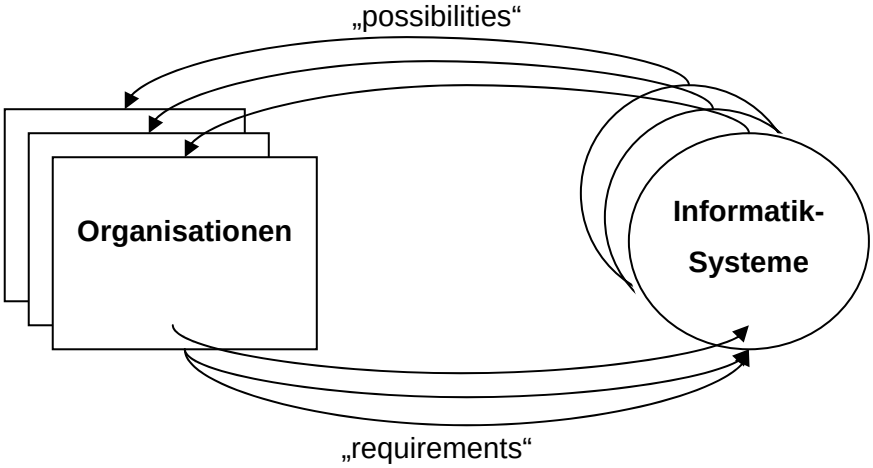

Abbildung 3-1: Überblick über den ISO/OWI-Orientierungsrahmen. Nach: ([Rolf 2002], S.53ff)

Die Abbildung 3-1 bietet einen Überblick zu dem ISO/OWI-Orientierungsrahmen. Wechselwirkung und Rückkopplungen zwischen Informatikssysteme und Organisationen können mit ihm beschrieben werden. Hiefür werden ökonomische, sozial- und arbeitswissenschaftliche Erkenntnisse mit einbezogen [Rolf 2002].

Die Möglichkeiten („possibilities“) der IT- Entwicklung stehen den Anforderungen („requirements“) der IT-Anwendung gegenüber. Die IT-Entwicklung ist eine Kooperation verschiedener Akteure. Die Akteure kommen aus verschiedenen Bereichen, z.B. der Informatik, der Politik, der IT-Herstellung, der Netz- & Dienstbetreiber, sowie aus unterschiedlichsten Verbänden.
Die Software-Entwicklung ist auf Seiten der Organisation anzusiedeln. Diese findet dort innerhalb eines Prozesses in Projekten statt. Die hier beteiligten Akteure sind System-Entwickler, Benutzer, Betriebsrat, Management unw. Diese Akteure müssen kooperieren, so dass auf dieser Seite Konflikte entstehen können [Rolf 2002].

Der ISO/OWI-Orientierungsrahmen stellt die Berührungspunkte zwischen der Informatik und der Gesellschaft dar und verwendet hierfür dir folgenden drei Ebenen:

- Makro-Ebene
- Meso-Ebene und die
- Mikro-Ebene

Er bietet auf den drei Ebenen eine Zuordnung sämtlicher beteiligter Akteure und eine Hilfestellung bei der Abschätzung gesellschaftlicher Folgen, leider fehlt ein Gestaltungsmittel [Rolf 2002].

Der Name MICROPOLIS-Modell ist eine sprachliche Änderung. MICROPOLIS setzt sich aus den Wörtern „Microelectronic“ und „polis“ [altgriech. Stadtstaat] zusammen und stellt eine Metapher für fachliches Handeln im gesamtgesellschaftlichen Kontext dar [Rolf 2004b]. Die Mikroelektronik befasst sich mit der mit dem Entwurf und der Herstellung von integrierten Schaltungen [Duden 1990].

Die Ebenen werden im MICROPOLIS-Modell Perspektiven genannt, da dieser Begriff die Funktion besser beschreibt. Tatsächlich werden die Akteure in dem Orientierungsrahmen aus drei Sichten bzw. Perspektiven mit unterschiedlichen Detaillierungsgrad und Fokus betrachtet [Rolf 2004a].

Das MICROPOLIS-Modell stellt das Orientierungswissen der Informationstechnik und der globalen Gesellschaft in einen Zusammenhang und versucht, den Kern der Informatik, im Kontext zur Wirtschaftsinformatik, zu erörtern.

Dadurch, dass das MICROPOLIS-Modell die Informatik in einen gesamtgesellschaftlichen Zusammenhang stellt, soll Orientierungswissen geschaffen werden.
Zudem bietet das Modell eine Integration von Orientierungs- und Verfügungswissen der Informatik. Es umfasst, ähnlich wie der vorangegangene Orientierungsrahmen, die Mikro-, Meso und die Makroperspektive (siehe Abbildung 3-2).

Die Mikro-Perspektive versucht, ohne technisch zu antworten, die Frage zu erörtern, was der Kern des Computers ist. Sie zeigt die Destruktion von Überlieferten und gleichzeitiger Konstruktion von Neuerungen durch den Einsatz von Rechnern auf.
Die Mikro-Perspektive stellt die technikimmanente Perspektive dar. Das Orientierungswissen kann durch die anderen beiden Perspektiven auf der Meso- und Makroebene betrachtet werden [Rolf 2004a].

Die Meso-Perspektive hat den Fokus der Einführung von Soft- und Hardware, sowie deren Forschung und Entwicklung. Die Perspektive richtet sich auf Kontexte und Bedingungen in den Organisationen [Rolf 2004b].

Die Makroperspektive bezieht das gesellschaftliche Umfeld, d.h. Regelungen und Institutionen der Gesellschaft in den Blickwinkel ein.

Diese Arbeit legt ihren Schwerpunkt auf die mittlere Ebene, der Meso-Perspektive, weil die Einordnung von softwaretechnischen Ansätzen auf dieser Ebene relevant ist (siehe Kapitel 5.1). Um diese im Gesamtkontext zu verstehen, folgt eine Erläuterung der drei Perspektiven, mit den Zusammenhängen und Anknüpfungspunkten untereinander.

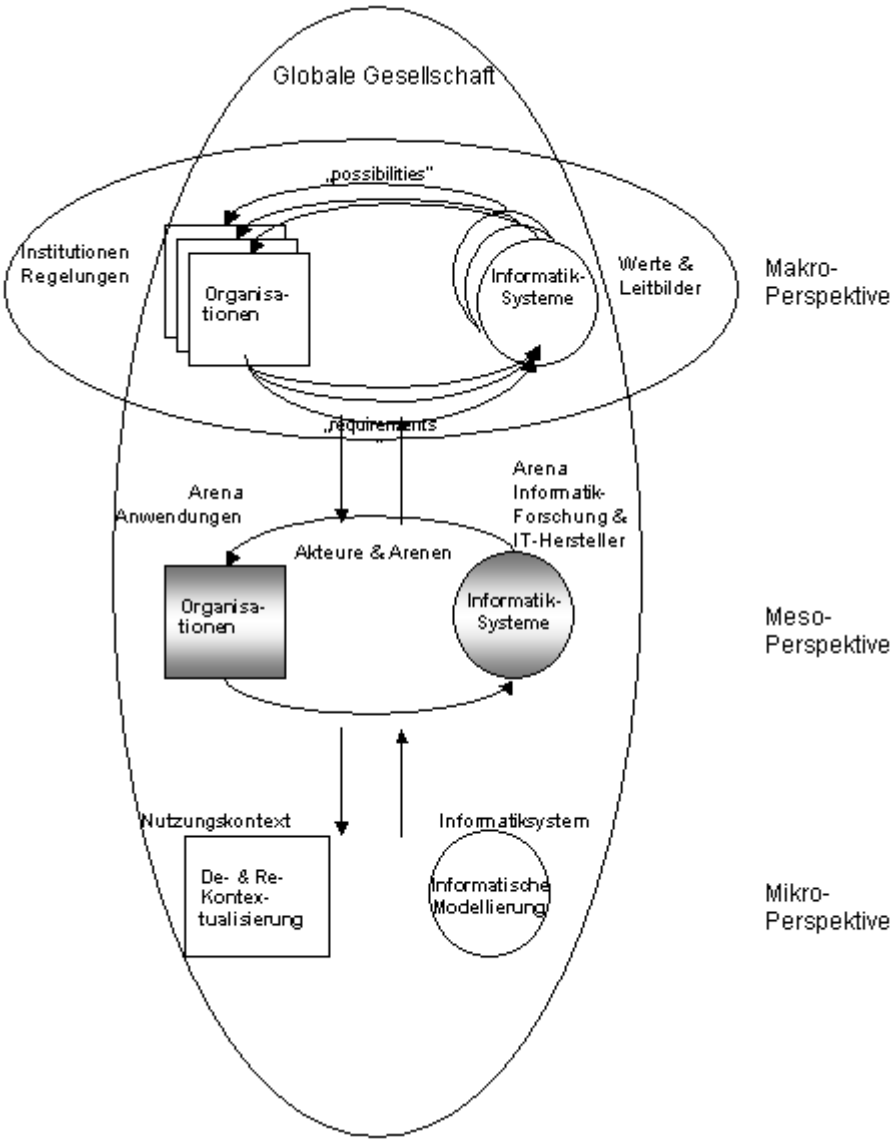

Abbildung 3-2: Das MICROPOLIS-Modell im Überblick [Rolf 2004b]

Im Folgenden beschreibe ich die einzelnen Perspektiven detaillierter.

3.1 Mikro-Perspektive

Die Mikro-Perspektive befasst sich mit dem „soziotechnischen Kern“ des Computers. Wie oben bereits erläutert, wird auf dieser Perspektive die Frage nicht technische beantwortet. Ferner betrachtet die Mikro-Perspektive das Verhalten einzelner Akteure, bzw. kleinerer Gruppen. Das Hauptaugenmerk liegt auf der Entwicklung von Systemen. Hier wird die Kommunikation zwischen Akteuren zur Konfliktbereinigung und zur Zielerreichung genutzt [Rolf 2004b].

Für den Kern der Informatik wird oft „Dijkstras Brandmauer der Informatik (siehe Abbildung 3-3)“ als Beispiel heran gezogen. Auf der Mikroperspektive stehen auf der einen Seite die Anwendungskontexte und dem gegenüber die Informatik im Bereich Forschung & Entwicklung, als auch die Lehre.

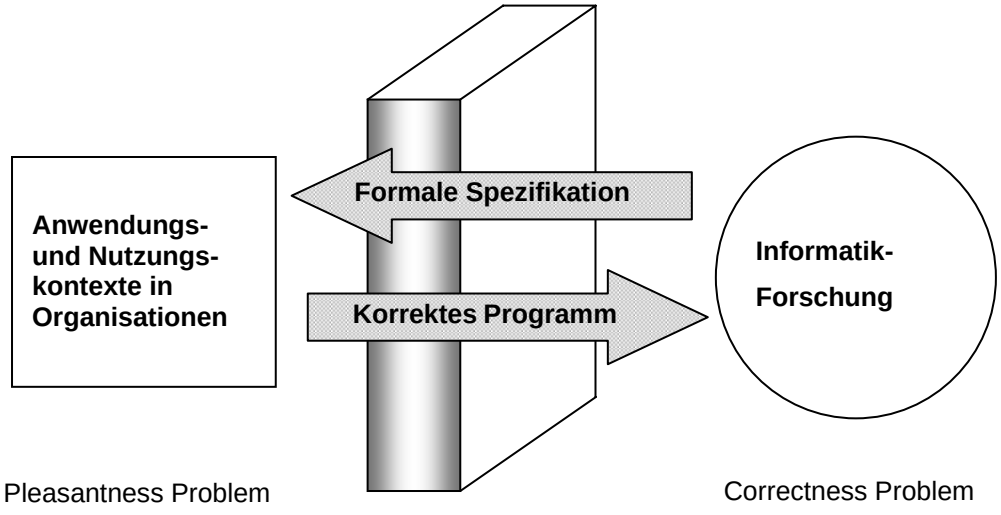

Abbildung 3-3: Dijkstras "Brandmauer der Informatik" [Rolf 2004b], S.10

Die Mikro-Perspektive versucht diese Mauer zu überwinden, nach der Informatiker nicht mit den Anforderungen der Anwenderpraxis befasst sein sollten [Rolf 2004b].

In Abbildung 3-2 zeigt auf der Mikro-Perspektive die Bereiche der Nutzungskontexte auf der einen Seite und die Informatiksysteme auf der anderen Seite. Im Wesentlichen beruht die Entwicklung der Informationstechnik auf dem informatischen Modellieren.

Ein Rechner führt Operationen aus und Methoden beschleunigen die auszuführenden Operationen bei der Entwicklung der Informationstechnik. Programme sind komplexe Verknüpfungen der Operationen. Die Möglichkeit beliebiger Kombination der Elemente schafft die Komplexität [Rolf 2004a].

Dekontextualisierung und Rekontextualisierung

Der Einsatz von Computern ersetzt Routinevorgänge durch maschinelle Abläufe. Dies verändert Prozesse in Organisationen.
Bei der Dekontextualisierung wird eine Handlung aus seinem individuellen Kontext herausgelöst und in ein formales, kontextfreies Modell übertragen. Die Rekontextualisierung entsteht bei der Ausführung durch den Rechner. Dann erst befindet sich die Handlung wieder in einem Kontext. Im Zentrum der Dekontextualisierung und Rekontextualisierung stehen das Verhalten und die Reaktionen von den Benutzern der Systeme. Die Softwaretechnik und die Software-Ergonomie berücksichtigen in ihren Modellen den Benutzer des Softwaresystems und beschäftigen sich mit den Fragen, wie sich Handlungen und Arbeitskontexte durch den Einsatz von Informationstechnik verändern. Dies ist ein Weg über Djikstras Brandmauer. Die Dekontextualisierung und die Rekontextualisierung verdeutlichen dies [Rolf 2004a]:

3.2 Meso-Perspektive

Die Meso-Perspektive hat ihren Fokus auf der Technikgestaltung und den Innovationen in Organisationen im Informatiksystem. Mit dieser Perspektive sollen Nutzungskontexte in Organisationen beschrieben werden, unter Einbeziehung der Abläufe im Informatiksystem und aus dem Blickwinkel der Dekonstruktion.
Dekonstruktion und Konstruktion sind bei technischen Neuerungen immer vorhanden. Durch Dekonstruktion, d.h. Abbauen, Abtragen bzw. „Auf-die-Seite-Stellen“ des Alten wird immer eine Neuerung konstruiert [Rolf 2004b].

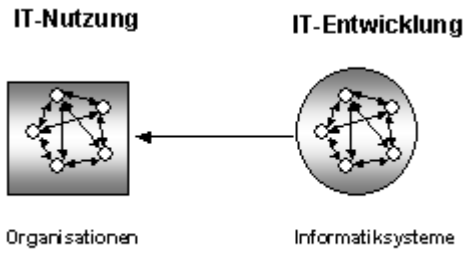

Abbildung 3-4: Das Akteursmodell mit Blick auf Organisationen und Informatiksysteme [Rolf 2004b], S.21

Die Softwareentwicklung ist in Ihrem Prozess abhängig von verschiedenen Akteuren, die unterschiedliche Interessen vertreten. Akteure sind Einzelpersonen, Personengruppen oder auch Gemeinschaften von Menschen mit gleichen Wertvorstellungen und Interessen.
Im Prozess der Softwareentwicklung sollten die verschiedenen Akteure berücksichtigt werden, da die Software den Akteuren als ein Werkzeug dienen soll [Rolf 1998].

Die Meso-Perspektive betrachtet Gruppen von Akteuren, die in Hierarchien und Prozessen geordnet sind. Dies gilt vor allem für Organisationen und Unternehmen.
Wichtig sind hierbei die Wechselwirkungen, die bei der Entwicklung von Organisationen und Informatiksystemen auftreten, zu verstehen. Die Meso-Perspektive bietet eine Möglichkeit Innovationen nachzuvollziehen und zu erklären [Rolf 2002].

Abbildung 3-4 zeigt das Akteursmodell mit Blick auf Organisationen und Informatiksystemen. Auf der einen Seite ist die Organisation und damit die IT-Nutzung angesiedelt, auf der anderen die IT-Entwicklung der Informatiksysteme.
Die IT-Entwicklung ist eine Kooperation verschiedener Akteure. Die Akteure kommen aus verschiedenen Bereichen, z.B. der Informatik, Politik, IT-Herstellung, Netz- & Dienstbetreibung, sowie aus Gemeinschaften oder Verbänden [Rolf 1998], [Rolf 2002].

Die Software-Entwicklung ist auf Seiten der Organisation anzusiedeln. Diese findet dort innerhalb eines Prozesses in Projekten statt. Die hier beteiligten Akteure sind System-Entwickler, Benutzer, Betriebsrat, Management usw. Diese Akteure müssen kooperieren, aber es können naturgemäß Probleme auf dieser Seite entstehen. Die Probleme sind Interessenkonflikte, Kommunikationslücken, u.ä. die bei Gemeinschaftsarbeiten auftreten können.

Das MICROPOLIS bietet hier eine Zuordnung sämtlicher beteiligter Akteure und eine Hilfestellung bei der Abschätzung gesellschaftlicher Folgen, leider fehlt ein adäquates Gestaltungsmittel [Rolf 2002].

Akteure und Arenen

Akteure sind von Interessen und Wertvorstellungen geleitete Einzelpersonen, Gruppen oder Gemeinschaften. Arenen sind Handlungsfelder in denen Akteure agieren.
Es gibt verschiedene Typen von Arenen:

- Arena Anwendungen
- Arena Informatik-Forschung und Gesellschaft

Die Arena Anwendungen betrachtet die Akteure im Anwendungskontext, d.h. in den Organisationen. Die Arena Informatik-Forschung und Gesellschaft betrachtet das Informatiksystem. Hier sind Hersteller von Informationstechniken, sowie die Informatik an Universitäten und in Forschungsinstituten anzusiedeln. Die Kommunikation der Akteure untereinander geht weit über die Grenzen der Arenen hinaus. Aus diesem Grund sind die Arenen nicht isoliert zu betrachten [Rolf 2004a].

Leitbilder und Methapern

Werte und Interessen von Akteuren drücken sich in Leitbildern und Metaphern aus. Die Leitbilder beinhalten Werte, Orientierungen und Sinn [Rolf 1998].
Unter Metaphern werden Bildhafte Ausdrücke verstanden, die aus ihrem Kontext heraus in einem neuen Zusammenhang verwendet werden [Züllighoven 1998].
Maschinen- und Systemorientierte Metaphern füllen die Sprache der Informatik. Sie ermöglichen durch Assoziationen einen sprachlichen Vergleich. Leitbilder und Metaphern dienen den Akteuren zur Kommunikation [Rolf 2004a].

Handlungen und Strukturen

Eine Handlung ist ein bewusster oder auch unbewusster Eingriff von Akteuren in die soziale Welt. Dieser Eingriff kann verändernd oder stabilisierend sein.
Strukturen sind dauerhafte Gegebenheiten einer Institution, wie z.B. Ressourcen oder Regeln. Die Struktur nach der ein Akteur handelt, ermöglicht die Handlung und beschränkt diese sogleich. Jede Handlung unterliegt einer Struktur [Rolf 2004a].

3.3 Makro-Perspektive

In der Makro-Perspektive spielen die gesellschaftlichen Einflüsse, die auf eine Entwicklung wirken eine wichtige Rolle. Gesellschaftliche Einflüsse setzen Trends und geben so die Richtung der Entwicklung vor. In der freien Marktwirtschaft werden keine Produkte entwickelt ohne eine Marktanalyse zur Akzeptanz dieser Produkte durchzuführen [Rolf 2004a+b].

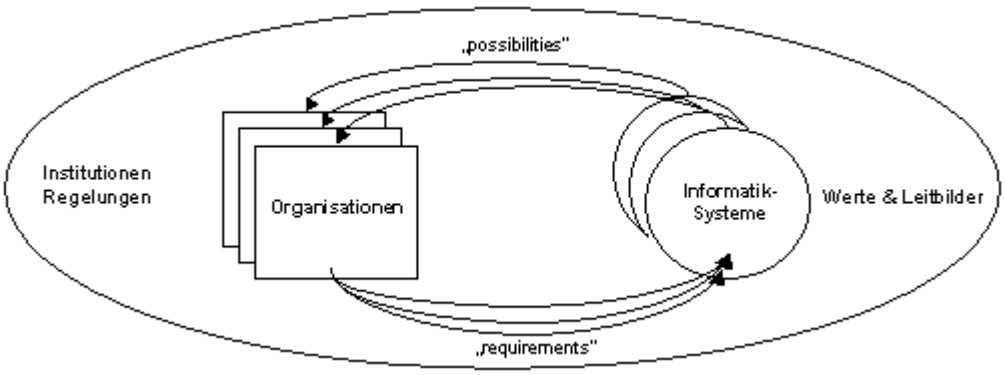

Abbildung 3-5: Makro-Perspektive [Rolf 2004b]

Die Makro-Perspektive beschreibt die Einbettung von Informatiksystemen und Organisationen in der Gesellschaft. Diese Perspektive setzt einen groben Rahmen für die unteren Perspektiven.
Die Einflussgrößen sind hierbei Werte, Traditionen, Gesetze, ökonomische Wertsetzungen usw. [Rolf 2004b].

Zwischen den Perspektiven entstehen Wechselwirkungen. Die Handlungen und Strukturen stehen in einem wechselseitigen Bedingungs- und Ermöglichungsverhältnis. Die gesellschaftlichen Strukturen sind das Resultat individueller Handlungsstrategien.
Die Makro-Perspektive setzt einen Rahmen für das individuelle Handeln auf den unteren Perspektiven.

Im Folgenden werden einige ausgewählte softwaretechnische Ansätze erläutert und beschrieben. Dann werden diese softwaretechnischen Ansätze im MICROPOLIS-Modell diskutiert und die in der Einleitung genannten Fragen erörtert.

Der Hauptfokus liegt auf der Meso-Perspektive des MICROPOLIS-Modells. Hier gibt es einige Anknüpfungspunkte zu der Softwaretechnik, da sich diese Perspektive, wie bereits dargestellt, mit der IT – Nutzung und der IT – Entwicklung befasst.

4 Ausgewählte softwaretechnische Ansätze

In diesem Kapitel werden verschiedene softwaretechnische Ansätze erläutert. Bevor dies getan werden kann, werden die für dieses Kapitel benötigten Begriffe beschrieben.

Hier wird kurz auf die beiden Fragen: „Was ist Softwaretechnik?“ und „Was ist Technik?“ eingegangen. Diese Begriffserklärungen sind wichtig für den Verlauf des Kapitels und damit für die Diskussion und die Schlussbetrachtung.

Was ist Softwaretechnik?

Theoretische Grundlagen, methodische Anleitung, technische Unterstützung, sowie die Organisation von Projekten zur Softwareentwicklung sind die Aufgaben der Softwaretechnik (engl. *software* engineering). Durch die „Softwarekrise“ 1968 entstand dieses Fachgebiet und hat sich bis heute etabliert.
Die damalige Sicht der Softwareentwicklung als Herstellung des Produktes Software trennte die Softwareherstellung von Softwareeinsatz. Laut Floyd eine eingeschränkte Sichtweise, da kooperative und kreative Anteile in der Softwareentwicklung nicht einbezogen wurden [Floyd 1995].

Die Design-Sicht betrachtet die Softwareentwicklung als Design von Artefakten und integriert technische, wirtschaftliche, soziale und ästhetische Gesichtspunkte. Die Software wird eingebettet in menschliches Arbeitshandeln. Der Lernprozess beinhaltet die Fusion von soziotechnischer Gestaltung und technischem Entwurf [Floyd 1994].

„Die Softwaretechnik befasst sich mit der professionellen Entwicklung großer Software-systeme, wobei Anwendungssoftware im Vordergrund steht.“ (siehe [Floyd und Züllighoven 1997], S. 642)

Grundlegend für die Softwaretechnik ist die Bereitstellung von Prinzipien, Methoden, Konzepten, Notationen und Werkzeugen. Die Bereitstellung sollte zielorientiert, d.h. unter Berücksichtigung von Kosten, Zeit und Qualität erfolgen. Die systematische Verwendung, sowie die Bereitstellung der softwaretechnischen Ansätze für die

arbeitsteilige, ingenieursmäßige Entwicklung und Anwendung von umfangreichen Software-Systemen ist die Aufgabe der Softwaretechnik. Zielorientiert umfasst die Berücksichtigung z.B. von Kosten, Zeit, Qualität. Die Forschungsrichtungen der Softwaretechnik sind:

- Modellierung für Analyse und Entwurf
- Werkzeuge für SW-Entwicklung und deren Verwaltung
- Software-Architekturen
- Konzepte für Programmierung und Spezifikation auf Anwendungsbereiche
- Projektorganisation und Qualitätssicherung [Floyd und Züllighoven 1997]

Was ist eine Technik ?

Eine Technik ist ein Verfahren zum Ausführen einer Methode.

> *„Unter Technik versteht man die allgemeine Kenntnis und Beherrschung der Mittel, die zur Ausübung eines Handwerks, einer Kunst und dgl. notwendig sind und die Handfertigkeit des Ausübenden (...).“* [Informatik-Begriffsnetz 2005].

Auf die Softwaretechnik übertragen gibt uns die Methode eine Richtlinie und die Technik das Mittel, diese Richtlinie umzusetzen. Eine Methode ist eine Vorgehensweise, wohingegen eine Technik ein konkretes Verfahren innerhalb einer Methode ist.

Erläuterung und Beschreibung verschiedener ausgewählter softwaretechnischer Ansätze

In diesem Abschnitt werden verschiedene softwaretechnische Ansätze, d.h. Methoden, Techniken, Modelle und Werkzeuge beschrieben und erläutert.
Primär wird darauf eingegangen, um welchen softwaretechnischen Ansatz es sich handelt (Methode, Technik, Werkzeug etc.) und was dieser Ansatz beinhaltet. Dabei werden die Kriterien des Ansatzes, sowie dessen Verwendung, ebenso wie die Relevanz beschrieben. Abschließend wird der ausgewählte softwaretechnische Ansatz dem MICROPOLIS-Modell zugeordnet.

Die ausgewählten softwaretechnischen Ansätze, die in diesem Kapitel beschrieben und erläutert werden, sind

- Objektorientierte Analyse (OOA)
- Anforderungsermittlung im Requirements Engineering

- Contextual Design (kontextuelles Design)
- Extreme Programming
- STEPS (Softwaretechnik für Evolutionäre Partizipative Softwareentwicklung)

4.1 OOA (Objektorientierte Analyse):

Die Objektorientierte Analyse ist eine spezielle Vorgehensweise und damit eine Methode in der Softwaretechnik, die den objektorientierten Entwurf in der Planung berücksichtigt. Wenn ein Softwareprojekt mit einer objektorientierten Sprache implementiert werden soll, ist es von besonderem Interesse dies bereits während der Planung zu berücksichtigen.
Die objektorientierte Analyse befasst sich mit den Gegenständen der Anwenderwelt, die in den Modellen abgebildet werden und sich so in späteren Systemen wieder finden.

- Statische Sicht: Klassen
- Dynamische Sicht: Objekte
- Akteurssicht: Anwendungsfälle (Use Cases [Schienmann 2002])

In die Objektorientierte Analyse ist die UML (Unified Modelling Language) integriert als eine graphische Darstellungsform.
Der Ursprung der UML war die „Unified Method 0.8“, die 1994 von Grady Booch und Jim Rimbaugh entwickelt wurde. Methoden, die sich in der Praxis bewährten, wurden zu einem Industriestandard weiterentwickelt. Seit 1995 wurde die UML mittels Methoden aus der Objektorientierten Softwareentwicklung durch Ivar Jacobsen erweitert. 1996 wurde die UML veröffentlicht. Die Object Management Group (OMG) verabschiedete 1997 UML als Standard [Balzert 1999b].

Die oben genannten Anwendungsfälle lassen sich in sogenannten Use Cases darstellen. Einfache Symbole stehen hierbei für die Akteure und deren Tätigkeiten. Szenarien können so graphisch dargestellt werden und als Diskussionsgrundlage den Entwicklern, sowie den späteren Anwendern dienen.

- Use Case
- Klassendiagramme
- Zustandsdiagramme

[Balzert 1999a]

4.2 Anforderungsermittlung/ Anforderungsentwicklung im Requirements Engineering

Requirements Engineering (Anforderungsmanagement) ist ein Teilgebiet der Softwareentwicklung. Anforderungsermittlung ist eine der vielen Aufgaben im Requirements Engineering. Prototyping und Contextual Design sind Entwicklungsmethoden, die das Problem der Anforderungsermittlung auf ihre Art und Weise verdeutlichen.
Anhand von Methoden und Techniken werden im Requirements Engineering Anforderungen mit den dazugehörigen Informationen verwaltet und analysiert [Rupp 2002].

> *„Eine Anforderung ist eine Aussage der zu erfüllenden Eigenschaft oder zu erbringenden Leistung eines Produktes, eines Prozesses oder der am Prozess beteiligten Personen.“* (Siehe [Rupp 2002] Seite 183)

Der erste Schritt für die Modellierung ist die Anforderungsermittlung. Es gibt verschiedene Arten, die Anforderungen eines Systems zu ermitteln. Mittels Arbeitsplatzanalyse, Szenarien, graphischer Modellierung und eines Glossars wird ein Modell für die Anforderungen in einem künftigen System erstellt. Die Arbeitsplatzanalyse erfolgt durch Interviews und Arbeitsplatzbewertungen. Szenarien dienen der Aufgabenbeschreibung einer Rolle. Die Rollen werden durch die verschiedenen Arbeiten der künftigen Benutzer des Systems festgelegt und deren funktionelle Soll und Ist-Aufgaben erörtert und beschrieben. Die Graphische Modellierung mittels Use-Cases oder Kooperationsbilder dient der Einordnung des Einzelnen in eine Gruppe, die im Zusammenhang zum gesamten System steht. Sämtliche Dokumente der Anforderungsermittlung werden in der Fachsprache der Anwendung, d.h. der Benutzer geschrieben. Um Verständnisprobleme zwischen Entwicklern und Benutzern zu verringern bzw. verhindern wird ein entsprechendes Glossar in der Fachsprache der Anwender erstellt. Die Entwickler sind Autoren des Glossars und schreiben dies mit Feedback der Anwender, die als Kritiker fungieren [Floyd und Oberquelle 2003a+b].

Hier gibt es einen Zusammenhang zum MICROPOLIS-Modell, da dieses die unterschiedlichen Akteure fokussiert.

Szenarien beschreiben Aufgabenbereiche eines Benutzers. Eine graphische Modellierung, wie z.B. Use Cases [Schienmann 2002] dient der Ermittlung der Anforderungen, indem sie den Benutzern und Entwicklern optisch einen Überblick liefern. Das Glossar erläutert die Fachbegriffe der Anwenderwelt und gibt dem Entwickler eine Hilfestellung diese zu begreifen.

Im Requirements Engineering gibt es drei unterschiedliche Arten von Anforderungen, funktionale, nicht-funktionale und domainspezifische. Die nicht-funktionalen werden unterschieden in Produktanforderungen, Organisationsanforderungen, sowie in externe Anforderungen [Rupp 2002].

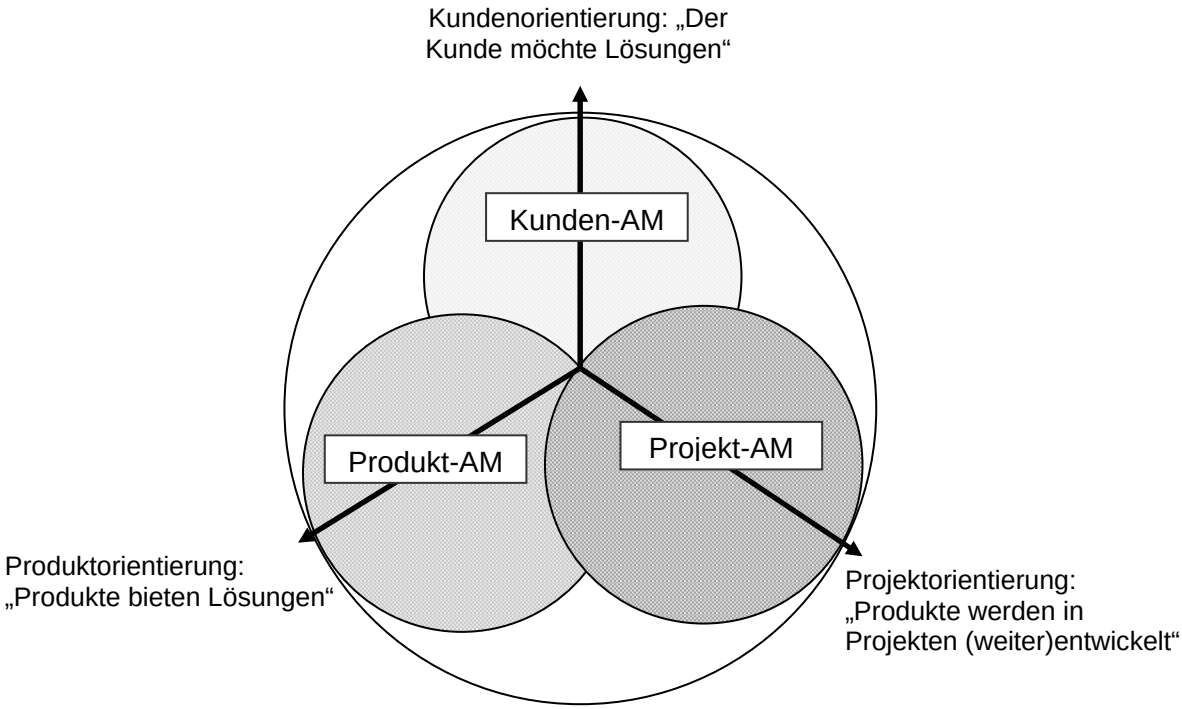

Abbildung 4-1: Die drei Prozessbereiche des Anforderungsmanagements [Schienmann 2002], S.17

Das Anforderungsmanagement, kurz AM geht auf die drei Sichten Kunde, Produkt und Projekt ein (siehe Abbildung 4-1). *Kundenorientierung* bedeutet, dass die Anforderungen vom Kunden formuliert werden. Diese müssen weder konkrete Projekte oder Produkte beinhalten. Die *Produktorientierung* bietet Lösungen für die Bedürfnisse der Kunden an. Sie werden anhand von Kundenanforderungen spezifiziert und dann in Projekte umgesetzt. In der *Projektorientierung* werden in Anwendungsentwicklungsprojekten die Produkte (weiter)entwickelt. Die Projekte sind

zeitlich begrenzt und mit einer festen Zielsetzung versehen. Das Ergebnis der Projekte sind die Problemlösungen für den Kunden.

Diese drei Sichten des Anforderungsmanagement führen zu den zentralen Prozess- bzw. Aufgabenbereichen:

- Kunden-Anforderungsmanagement (Kunden-AM)
- Produkt-Anforderungsmanagement (Produkt-AM)
- Projekt-Anforderungsmanagement (Projekt-AM)

Im Kunden-AM werden in der Systementwicklung die Kundenbedürfnisse berücksichtigt und sichergestellt, dass diese auch in Produkte umgesetzt werden.

Das Produkte-AM ist in der Produktentwicklung verantwortlich für dessen Profitabilität und Nachhaltigkeit.

Die Produktanforderungen werden in der Projekt-AM verfeinert und umgesetzt. Dies geschieht unter Einhaltung der gesetzlichen Rahmenbedingungen [Schienmann 2002].

In diesen drei Aufgabenbereichen werden unterschiedliche Anforderungen betrachtet. Jeder Bereich beschäftigt sich somit mit unterschiedlichen Ausrichtungen, Techniken und Methoden. Weit verbreitete Techniken zur Anforderungsermittlung sind Interviews, Fragebögen, Prototyping, Beobachtung von Mitarbeitern und Anforderungsworkshops. Die ermittelten Anforderungen werden inhaltlich und formal in der Anforderungsanalyse ermittelt. Begriffs- oder Bedeutungsanalysen, Anwendungsfälle, Problem Frames, CRC-Karten, sind Techniken die der Anforderungsanalyse dienen.

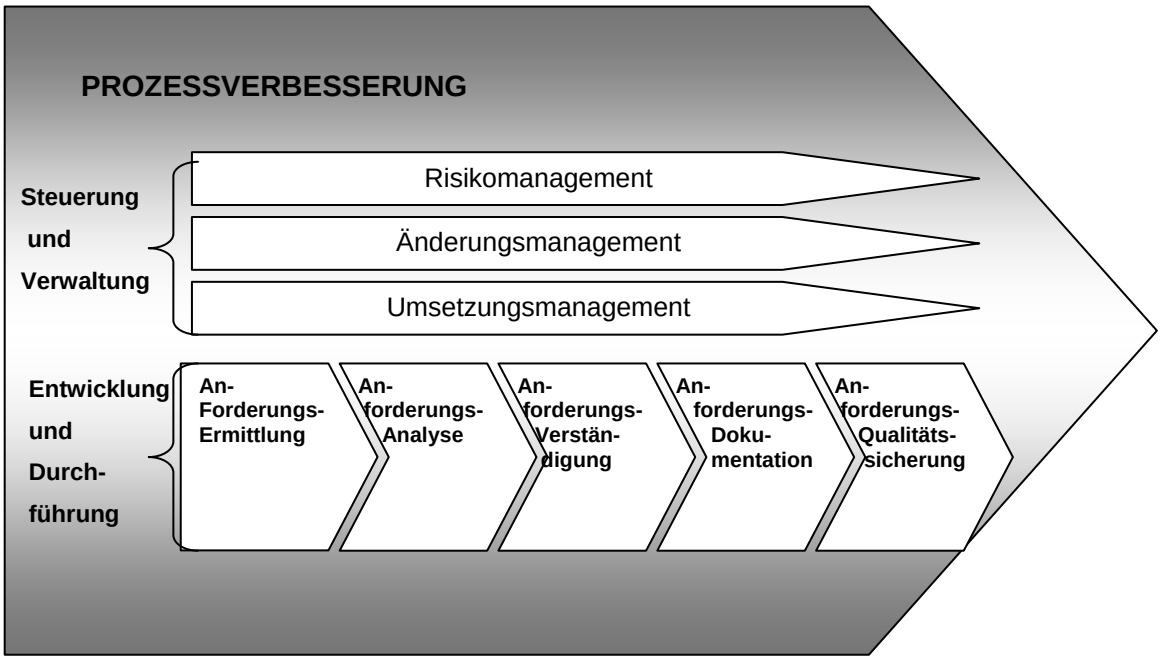

Abbildung 4-2: Hauptaufgaben im Anforderungsmanagement [Schienmann 2002], S.33

In der Abbildung 4-2 werden die Aufgaben und Aktivitäten des Anforderungsmanagement dargestellt. Die Prozessverbesserung beinhaltet sowohl den operativen, als auch funktionalen Umgang mit dem Anforderungsmanagement. Die Aufgaben werden nicht zwangsläufig hintereinander ausgeführt, sondern können ineinander greifen.

Operative Aufgaben (Steuerung und Verwaltung)

- Risikomangement: Risiken erkennen und entsprechende Lösungen finden, um diese zu vermindern.
- Änderungsmanagement: Kontrollierter Umgang mit Änderungen.
- Umsetzungsmanagement: Planung und Kontrolle des Entwicklungsprozesses.

Funktionale Aufgaben (Entwicklung und Durchführung)

- Anforderungsermittlung: Anforderungen, Empfehlungen, Wünsche, sowie Randbedingungen werden erhoben.
- Anforderungsanalyse: Konkretisierung und fachliche Klärung der ermittelten Anforderungen.
- Anforderungsverständigung: Einscheidung und Einigung über Anforderungen.
- Anforderungsdokumentation: Strukturierung der Anforderungen und spezifizieren der Beschreibungsmuster.
- Anforderungsqualitätssicherung: Sicherung der formalen Qualität der Anforderungen und inhaltliche Verifikation und Validierung.

[Schienmann 2002]

4.3 Contextual Design

Zentraler Punkt dieser Gestaltungsmethodik ist das Umfeld (der Kontext) einer Arbeit. Es wird dafür die Frage nach der Relevanz für die Arbeit gestellt. Im Mittelpunkt stehen die Struktur und die Art der Nutzung eines Systems.

Die Grundlegenden Fragen im Contextual Design sind:

- Was ist für die Arbeit relevant?
- Wie soll das neue System genutzt werden können?
- Wie soll das System strukturiert sein?
- Ist das gebaute System nutzbar?

Die Methode ist ganzheitlich und arbeitsorientiert. Im Contextual Design ist die Vorgehensweise eine von Außen nach Innen gehende (siehe Abbildung 4-3). Erst wird

das Umfeld betrachtet, dann Arbeit, danach welche Funktionen notwendig sind und im letzten Schritt werden die Interaktionen betrachtet [Floyd und Oberquelle 2003a], VL8.

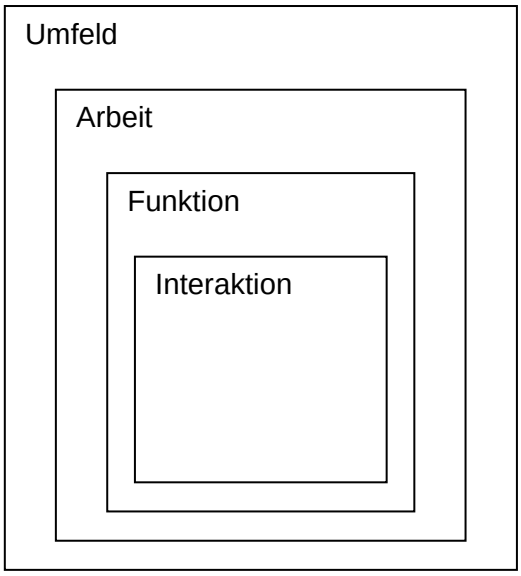

Abbildung 4-3 Vorgehensweise im Contextual Design Nach: [Floyd und Oberquelle 2003a], VL8

Der benutzerorientierte Designprozess des Contextual Design hilft dem Entwickler-Team die Anforderungen des Kunden zu erkennen und das System entsprechend zu gestalten [Beyer und Holzblatt 1998].

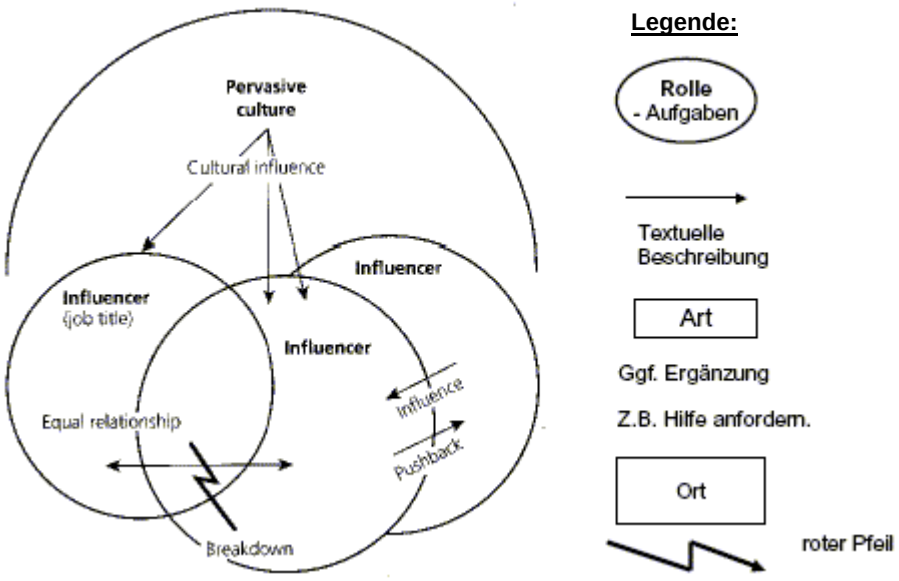

Abbildung 4-4: Einfluss-Modellierung (cultural modelling) Nach: [Floyd und Oberquelle 2003a]

Beim Softwaredesign gilt es zunächst, die Benutzer zu verstehen – die Wichtigkeit der Themen und Aufgaben, die Gemeinschaftsprojekte, den Informationsbedarf und die Werte. Hierzu werden über Befragungen der Benutzer die Benutzerdaten aus der Praxis erfasst. In der Konsolidierungsphase trägt das Entwickler-Team Daten aus den einzelnen Benutzerbefragungen zusammen, so dass es übergreifende Muster und Strukturen erkennt. Auf diese Weise können auch individuelle Abweichungen berücksichtigen werden. Als Ergebnis dieser Phase wird ein geschlossenes Bild der Benutzer deutlich. Auf Flipcharts werden Visionen des neuen Systems graphisch aufgezeichnet, um die richtigen Schwerpunkte zu setzten. Die Vision dient dem Entwickler-Team als Leitfaden. Bei dem Design der Oberfläche werden intuitive Bedienbarkeit und Benutzerfreundlichkeit in den Vordergrund gesetzt. Um frühzeitig mögliche Probleme bei der Systemkonstruktion zu erkennen, wird der Prototyp des Systems auf Papier dargestellt [Holzblatt 1999].

In Abbildung 4-4, einer typischen Graphik des Contextual Design, werden die verschiedenen Einflussfaktoren, die auf ein Unternehmen einwirken dargestellt. Personen oder Gruppen nehmen Einfluss auf die Firmenkultur, haben Erwartungen, Wünsche und liefern das Firmenprofil. Die Einflüsse sind Standards in der Firmenpolitik, die Macht verschiedener Personen, das Wertesystem, Emotionen, individuelle Stile und Werte, sowie die Vorlieben des Einzelnen oder der Gruppen [Floyd und Oberquelle 2003a].

Mit einfachen Graphischen Mitteln, d.h. Kreisen, Rechtecken, Pfeilen und textuellen Beschreibungen werden im Contextual Design die unterschiedlichen Einflüsse, Ziele und Visionen modelliert und gestaltet.

4.4 eXtreme Programming

eXtreme Programming (XP) ist eine agile (lat. flink, beweglich, wendig) Methode, die sich wie alle agilen Methoden durch die folgenden Eigenschaften auszeichnet:

- Prozess mit wenigen Dokumenten
- Prozess steht regelmäßig zur Diskussion
- Collective Code Ownership (siehe unten „zwölf Techniken“)
- Kunde ist Teil des Entwurfsprozesses
- Zyklen sind von kurzer Dauer

[Floyd und Oberquelle 2003b], VL16

Erstmals wurde diese spezielle Entwicklungsmethodik (Entwicklungsprozess) von Beck im Jahre 2000 beschrieben [Beck 2000]. Sie unterliegt den vier Werten Einfachheit (engl. Simplicity), Kommunikation (engl. Communication), Feedback (engl. Feedback) und Mut (engl. Courage). Diese Werte sollte jeder, der eXtreme Programming einsetzt, verfolgen. Weiter baut eXtreme Programming auf Prinzipien auf, die dem Handeln im Projekt Richtlinien bieten. Es gibt zwölf Prinzipien, die fünf zentralen sind:

- Unmittelbares Feedback (engl. Rapid feedback)
- Einfachheit Anstreben (engl. Assume Simplicity)
- Inkrementelle Veränderung (engl. Incremental Change)
- Qualitätsarbeit (engl. Quality Work)

Unmittelbares Feedback zwischen den Entwicklern und den späteren Anwendern des Systems vermeidet Missverständnisse. Einfachheit anstreben (KISS: Keep It Simple, Stupid) bedeutet, dass das System so einfach wie möglich aufgebaut werden soll, so dass spätere Erweiterungen ebenfalls einfach umgesetzt werden können. Ausgehend von einem einfachen lauffähigen Grundsystem, wird das Endgültige durch Inkrementelle Veränderung nach und nach erweitert. Wichtig dabei ist, dass die Qualität bei der Entwicklung nicht aus den Augen verloren wird [Lippert et al. 2002].

Die zwölf Techniken des eXtreme Programming

Aus den Prinzipien wurden zwölf Techniken (engl. Practice) entwickelt, die die Umsetzung der Prinzipien unterstützen:

- *Kunde vor Ort* (engl. On-Site Customer): Der Kunde wird als Mitarbeiter in das Projekt eingesetzt. Er vermittelt in Gesprächen die Anforderungen an das System und gibt Feedback.
- *Planungsspiel* (engl. Planning): Entwickler und Kunde planen gemeinsam die nächsten Iterationen bzw. Relaeses der Software. Diese werden auf Karteikarten (siehe Story-Cards, Task-Cards) notiert. Der Entwickler schätzt den Entwicklungsaufwand und prioriesiert die einzelnen Karteikarten.
- *Metapher* (engl. Metapher): Hinter einem Softwareprojekt sollte eine Metapher stehen, die allem Beteiligten ein Bild über die Systemarchitektur vermittelt und die Richtlinien bestimmt. Sie beinhaltet beispielsweise welche Annahmen über

ein System gemacht werden oder wie es funktioniert. Wichtig hierbei ist, dass die zugrundeliegende Metapher allen beteiligten Personen geläufig ist.

- *kurze Releasezyklen* (engl. Short Releases): Iterationen in kurzen Zeitspannen dienen dazu möglichst kundenorientiert zu entwickeln und dem Kunden lauffähige Versionen zur Verfügung zu stellen. Der Vorteil hierbei ist es Mängel zeitnah beheben zu können.
- *Testen* (engl. Testing): Komponenten und Akzeptanztests bzw. Funktionstests sind die zwei wesentlichen Testarten im eXtreme Programming. Komponententests werden vom Entwickler ausgeführt und dienen dem zur Fehlerbehebung im Programmcode. Akzeptanztests werden anhand vom Kunden beschriebener konkreter Situationen durchgeführt. Sie geben einen Einblick für den Systemeinsatz.
- *Einfaches Desi*gn (engl. Simple Design): Das Systemdesign soll möglichst so einfach und auf das nötigste beschränkt sein, damit das System leicht verstanden, schnell entwickelt und effektiv weiterentwickelt werden kann. Es wird daher beim XP auf ein „Big Upfont Design“ („Design auf Vorrat“) verzichtet.
- *Refactoring* (engl. Refactoring): Unter Refactoring wird die Umstrukturierung eines Systems und Beibehaltung der Funktionalität verstanden. Das Refactotring dient dazu, die Struktur eines Softwaresystems flexibel zu gestalten. Es bietet somit den Vorteil ein Softwaresystem mit wenig Aufwand an aktuelle Bedürfnisse anzupassen. Komponententests werden beim Refactoring verwendet, um Seiteneffekte auszuschließen.
- *Programmieren in Paaren* (engl. Pair Programming): Die Entwickler programmieren in Paaren, in denen ständig zwei Entwickler gemeinsam an einem Entwicklungsrechner sitzen und gemeinsam ohne jegliche Aufgabenteilung arbeiten. Die Grundidee liegt in einem ständigen Review auf der Code- und der Design-Ebene. Dies dient einer hohen Qualität auf den genannten Ebenen. Die Paare wechseln den Arbeitspartner innerhalb der gesamten Entwickler eines Projektes, so dass jeder Entwickler jeden Teil des Codes kennen lernt.
- *Gemeinsame Verantwortung* (engl. Collective Ownership): Das gesamte Projektteam ist für ein Projekt gemeinsam verantwortlich. Dies gilt auch für die Dokumente und die Quelltexte. Jedes Projektmitglied darf und kann jeder Zeit alles ändern. Wenn ein Projektmitglied nicht anwesend ist, kann jeder andere einspringen. Dies setzt eine ähnliche Qualifikation der beteiligten Personen voraus.

- *Fortlaufende Integration* (engl. Continuous Integration): Durch die gemeinsame Verantwortung in XP-Projekten sind die Entwickler gleichberechtigt und können jeder Änderungen am Quelltext vornehmen, sofern er dies für notwendig hält. Um diese Änderung den anderen Entwicklern zugänglich zu machen, werden diese auf Integrationsrechner übertragen. Dieser Rechner ist einer, der ausschließlich für die Integrationen von neuen Quellen genutzt werden sollte. Vor jeder Intergration sollte der Quelltext getestet werden, so dass immer eine lauffähige und aktuelle Version des Gesamtsystems auf dem Integrationsrechner vorliegt.
- *Programmierstandards* (engl. Coding Standards): Die äußere Form eines Quelltextes wird durch Programmierstandards festgelegt.
- *40 Stunden Woche* (engl. 40-hours Week): Um die Leistungsfähigkeit der Entwickler und damit die Qualität in der Softwareentwicklung zu gewährleisten wird im eXtreme Programming für eine 40-Stunden plädiert. Eine dauerhafte Überlastung ist kontraproduktiv nach Ansicht des XP, da es zu minderer Leistungsfähigkeit und damit auch minderer Qualität führt. Dabei werden kurzfristige Mehrarbeiten in sogenannten „heißen Phasen“ nicht ausgeschlossen. Zudem werden die Programmierzeit, die Verwaltungszeit und die Pausen voneinander getrennt betrachtet.

([Beck 2000] , [Lippert et al. 2002])

Die Techniken werden mittels Artefakte durchgeführt. Es gibt vier Artefakte, die Handlungen und Routinen vergegenständlichen:

- Story-Cards: Auf Story-Cards, meist DIN-A5 große Karteikarten werden fachliche Anforderungen an das System aus Sicht des Kunden, dem späteren Anwender beschrieben. Die Story-Cards dienen dazu eine grobe Übersicht zu erhalten und können in Task-Cards unterteilt werden.
- Task-Cards: Zusätzlich zu der Aufteilung der Story-Card können die Task-Cards auch technische Anforderungen enthalten. Task-Cards unterscheiden sich also von Story-Cards, indem sie detailliertere Auskünfte über Aufgaben enthalten
- Releaseplan: Der Releaseplan ist ein Text oder eine Präsentation und enthält die Informationen, wann welche Release geplant sind. Die Entwicklungszeiten sind in kurze Zyklen eingeteilt, bei denen am Ende jedes Zykluses die Entwicklung des nächsten Releaseplanes steht. Bei größeren Projekten ist es notwendig einen Plan über das Gesamtprojekt zu erstellen. Dieser beschreibt

die einzelnen Projektetappen eines Kernsystems mit Ausbaustufen, sowie Spezialsystemen.

- Iterationssplan: In einem Release können mehrere Iterationen enthalten sein. Diese werden anhand von prioriesierten Story – und Task-Cards geplant [Lippert et al. 2002].

eXtreme Programming ist in zyklische Phasen organisiert. Die Anforderungsermittlung wird durch offene Interviews und Arbeitsplatzbeobachtungen mit dem Ergebnis von Szenarios und der Erstellung eines Glossars durchgeführt. Die Projektplanung berücksichtigt die verschiedenen Zielgruppen. Dabei werden Auftraggeber, Anwender und das Programm miteinbezogen [Lippert et al. 2002].

Beim eXtreme Programming steht der Kunde im Vordergrund der Betrachtung, so dass sich diese Methode gut auf der Meso-Perspektive des MICROPOLIS-Modells anordnen lässt. Die einzelnen Techniken, die im eXtreme Programming ausgeführt werden, sind so ausgerichtet, dass der Kunde in den Entwicklungsprozess mit einbezogen ist. Dies hat den Vorteil, das Missverständnisse frühzeitig erkannt und Fehler im Vorwege verhindert werden. Das MICROPOLIS-Modell betrachtet Konflikte, die (siehe Kapitel 3 MICROPOLIS-Modell) durch unterschiedliche Interessen der Akteure oder Kommunikationslücken entstehen können. Im Extreme Programming werden die unterschiedlichen Rollen und die einzelnen unterschiedlichen Interessen vertreten und miteinbezogen.

Andere agile Methoden sind z. B. Adaptive Software Development (ASD), Open Source Software Development (OSSD), The Rational Unified Process (RUP), Feature Driven Development (FDD), Crystal (family of methodologies) und Scrum [Floyd und Oberquelle 2003b], VL16.

4.5 STEPS (Softwaretechnik für Evolutionäre Partizipative Softwareentwicklung)

STEPS, Softwaretechnik für Evolutionäre Partizipative Softwareentwicklung ist ein theoretischer und methodischer Ansatz der Softwaretechnik, der an der Technischen Universität Berlin ausgearbeitet und erprobt wurde [Floyd et al. 1990].

STEPS eignet sich vor allem für Softwaresysteme, die in Arbeitszusammenhängen interaktiv benutzt werden [Floyd und Oberquelle 2003a+b].

In der Softwareentwicklung gibt es zwei unterschiedliche Sichtweisen, die produktorientierte und die prozessorientierte Sichtweise. Beide Sichtweisen stehen sich komplementär gegenüber.
Bei der produktorientierten Sichtweise wird Software als ein eigenständiges Produkt betrachtet. Dieses besteht aus Programmen und definierten Dokumenten. Diese Sichtweise trennt die Softwareentwicklung vom Einsatz der Software. Die Softwareentwicklung wird hier als Herstellung des Produktes Software angesehen.
Im Gegensatz dazu betrachtet die prozessorientierte Sichtweise die Software in dem Kontext von Prozessen. Diese Lern-, Arbeits- oder Kommunikationsprozesse bringen das Produkt Software hervor. Bei dieser Sichtweise werden die Herstellung und der Einsatz von Software gemeinsam und miteinander verschränkt betrachtet.
Das in Abbildung 4-5 dargestellte zyklische Projektmodell bezieht sich auf die produkt- und prozessbezogenen Aufgaben, die die unterschiedlichen Sichtweisen wiederspiegeln.
Koordination und Etablierung von Softwareentwicklungsprozessen sind prozessbezogene Aufgaben. Systemgestaltung und Softwarerealisierung sind produktbezogene Aufgaben.
In dem Modell wird das Produkt als eine Folge von Systemversionen und Dokumenten betrachtet. Der Prozess wird als eine Folge von Zyklen aufgefasst, die Herstellung und Einsatz beinhalten [Floyd 1992].

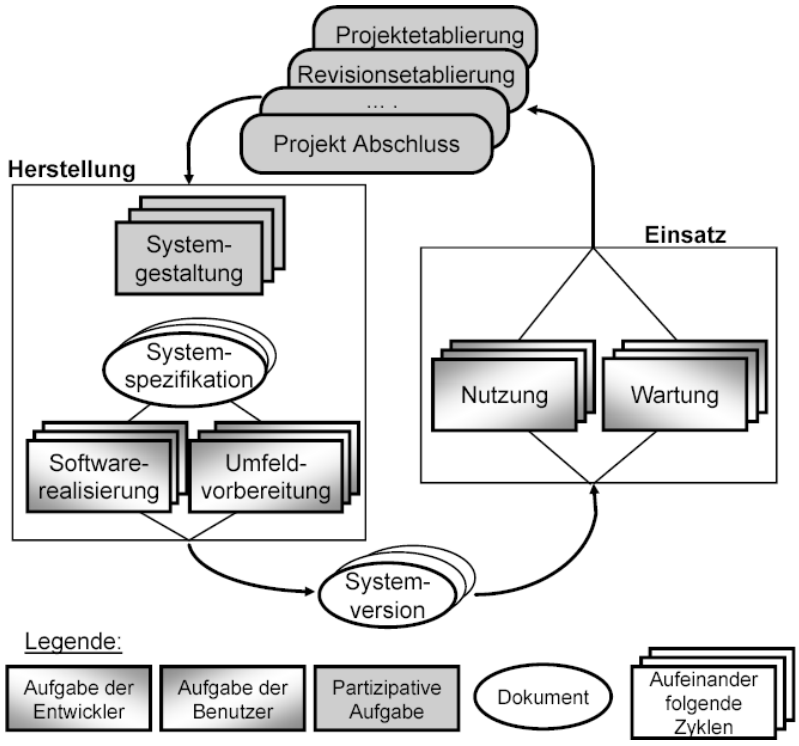

Abbildung 4-5 STEPS [Floyd und Oberquelle 2003a], VL4

Das Projektmodell

Die Systementwicklung wird als Folge von Entwicklungszyklen verstanden.
Jeder Zyklus hat die Herstellung und den Einsatz einer Systemversion zum Gegenstand. Das Versionskonzept trägt der evolutionären Veränderung der Anforderung der Benutzer und der Einsatzorganisation während des Einsatzes Rechnung [Floyd und Oberquelle 2003].
Innerhalb der Zyklen werden verschiedene Phasen durchlaufen und zu größeren Einheiten gruppiert. Abbildung 4-6 zeigt die partizipative und produktbezogene Aufgabe der Systemgestaltung, diese vereint die Phasen der Anforderungsermittlung und Systemgestaltung. Das Ergebnis dieser Einheit ist die Systemspezifikation [Floyd 1992].

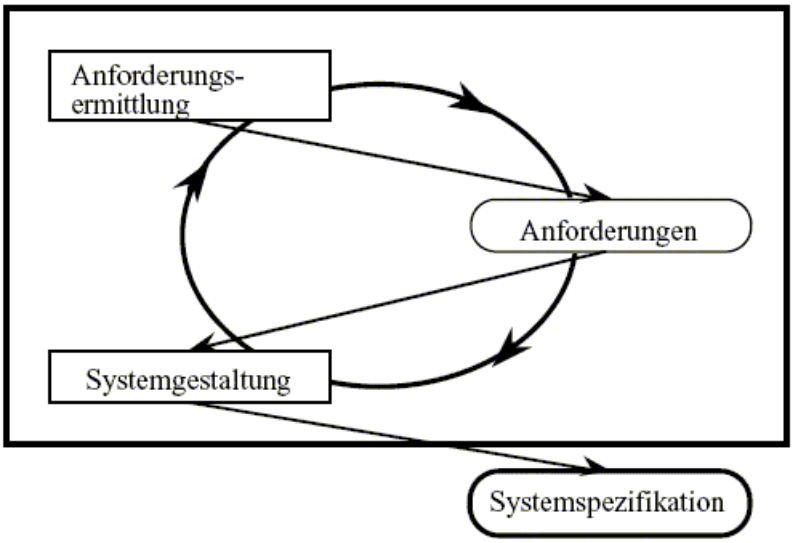

Abbildung 4-6: Zusammenhang zwischen Phasenmodell und zyklischem Modell: Systemgestaltung [Floyd 1992], S.15

Die Systemgestaltung:

Vier Prinzipien sind die Grundlage der Systemgestaltung.

- Ermittlung der Anforderungen
- Gestaltung der Nutzung
- exploratives Prototyping
- experimentelles Prototyping [Floyd und Oberquelle 2003]

Die Anforderungsermittlung ist eine Phase zur Festlegung der Anforderungen an die Software. Zur Ermittlung der Anforderungen gibt es Methoden, die Anforderungen erheben und Methoden die Anforderung beschreiben.
[Floyd 1992]

Unter einem Prototyp versteht man eine spezielle Ausprägung eines Systems [Floyd und Züllighoven 1997].
Es gibt unterschiedliche Arten von Prototyping, nämlich exploratives, experimentelles und evolutionäres Prototyping.
Das explorative Prototyping dient der Feststellung der Funktionalität eines späteren Systems.
Das experimentelle Prototyping als Demonstration von Lösungsvorschlägen, sowie zur Diskussionsgrundlage zwischen den Softwareentwicklern und den Benutzern. Es wird für unterschiedliche Zwecke eingesetzt. Der Einsatz kann eine vollständige oder teilweise Simulation des Systemmodells sein oder zur Demonstration von kritischen Entwurfsentscheidungen dienen. Experimentelles Prototyping kann zur Gestaltung der Benutzerschnittstelle genutzt werden oder zur Entwicklung eines Programmskeletts

dienen, bei dem die Funktionsweise des Systems im Ganzen erprobt werden kann durch die minimale Verknüpfung von Entwurfskomponenten.

Das evolutionäre Prototyping setzt ein inkrementelles Vorgehen bei der Softwarerealisierung voraus. Hier wird ein minimales Programmskelett implementiert, was dann nach und nach in Ausbaustufen weiterentwickelt wird und das spätere System ergibt [Floyd 1992].

5 Diskussion der Softwaretechnik im MICROPOLIS-Modell

In diesem Kapitel sollen die Softwaretechnischen Ansätze im MICROPOLIS-Modell diskutiert werden.

Hierbei ergeben sich zwei Fragestellungen:

1. Kann das MICROPOLIS-Modell Hinweise für softwaretechnische Fragestellungen geben?
2. Wie könnte eine Integration von STEPS und MICROPOLIS-Modell gelingen?

5.1 Kann das MICROPOLIS-Modell Hinweise für softwaretechnische Fragestellungen geben?

Um diese Fragestellungen beantworten zu können, müssen die Ansätze auf den verschiedenen Ebenen bzw. unter den verschiedenen Perspektiven betrachtet werden. Zudem muss betrachtet werden, wo die Anknüpfungspunkte zwischen dem MICROPOLIS-Modell und den ausgewählten softwaretechnischen Ansätzen liegen.

Die Frage 5.1 lässt sich beantworten durch

1. durch eine Bereicherung der softwaretechnischen Ansätzen durch das MICROPOLIS-Modell,
2. mit der Frage auf welchen Ebenen bzw. mit welchen Perspektiven des MICROPOLIS-Modells softwaretechnische Ansätze, im Speziellen der Ansatz STEPS zu verorten sind?

1. Bereicherung von softwaretechnischen Ansätzen:

Wie kann das MICROPOLIS-Modell die softwaretechnischen Ansätze bzw. die Softwaretechnik bereichern?

Arbeits- und Technikgestaltung fokussieren die Prozesse und lenken daher den Blick auf die Folgen und Wirkungen [Volpert 2002].

Das MICROPOLIS-Modell betrachtet die Akteure und Allianzen zwischen den Akteuren in den Blick. Konflikte, die in diesem Rahmen entstehen, können aus dieser Perspektive beobachtet werden. Diese Sichtweise kann der Softwaretechnik hilfreich sein, da das MICROPOLIS-Modell auf Zusammenhänge hinweist und so eine Orientierung der verschiedenen Ansätze bietet. Softwaretechnische Ansätze, die nach der Design-Sicht vorgehen, können vom MICROPOLIS-Modell betrachtet werden.

Wie in Kapitel 3 beschrieben, haben Leitbilder und Metaphern eine große Bedeutung in und für das MICROPOLIS-Modell. Welche Bedeutung haben Leitbilder in der Softwaretechnik? Was ist Leitbild und wofür dient es?

Um diese Frage zu beantworten, greife ich die Definition über Leitbilder von Züllighoven auf:

> *„Ein Leitbild ist eine benennbare, grundsätzliche Sichtweise, anhand derer wir einen Ausschnitt von Realität wahrnehmen, verstehen, gestalten. Ein Leitbild repräsentiert immer auch eine Wertvorstellung.“* (siehe [Züllighoven 1998], S-73)

Über diese Definition hinaus betrachtet Züllighoven das Leitbild im Kontext der Softwareentwicklung.
Das Leitbild stellt den gemeinsamen Orientierungsrahmen für die beteiligten Gruppen in einem Entwicklungsprozess. Es basiert auf Zielvorstellungen und Wertvorstellungen. Es unterstütz sowohl den Entwurf von Software, als auch die Verwendung und Bewertung. Ein Leitbild kann analytisch oder konstruktiv verwendet werden (siehe [Züllighoven 1998], S. 73).

Interessant ist, das Züllighoven in diesem Zusammenhang den Begriff „Orientierungsrahmen“ verwendet. Da das MICROPOLIS-Modell nach Rolf auch ein Orientierungsrahmen ist, ergibt sich hier eine Verbindung.
Ergänzend hierzu beschreibt Rolf, das Leitbilder für die Forschung und Lehre die Aufgabenfelder, Denkmuster und Schneisen vorgegeben (siehe [Rolf 1999], S.184).

Daraus resultiert, dass das MICROPOLIS-Modell ein Orientierungsrahmen und damit ein Leitbild ist. Der Orientierungsrahmen stellt somit die Aufgaben, Denkmuster und Schneisen bereit. Das MICROPOLIS-Modell betrachtet sowohl die Gesellschaft, als auch Informatiksysteme in Organisationen. Hieraus ergibt sich, dass dieser Orientierungsrahmen den Wirtschaftsinformatikern ein Leitbild bereitstellt.

Bereits 1992 fordert Volpert eine Erweiterung der Informatik in eine Gestaltungs- und Entwurfs-Wissenschaft mit der Hauptaufgabe der Organisations- und Arbeitsplatzgestaltung (siehe [Volpert 1992], S. 172).

In dem Orientierungsrahmen MICROPOLIS-Modell finden sich diese Ideen wieder, da das MICROPOLIS-Modell sowohl die einzelnen Akteure an ihrem Arbeitsplatz, als auch im Vergleich zu anderen Akteuren betrachtet (siehe Kapitel 3 MICROPOLIS-Modell).

2. Auf welchen Ebenen des MICROPOLIS-Modells sind softwaretechnische Ansätze im Speziellen der Ansatz STEPS zu verorten?

Für diese Frage werden die ausgewählten softwaretechnischen Ansätze auf den drei Ebenen betrachtet, um dann eine Zuordnung zu der jeweiligen zu treffen. Hierzu fasse ich die Aussagen der Ebenen aus Kapitel 3 noch einmal kurz zusammen und gebe entscheidende Anknüpfungspunkte wieder.

Soziale Gebilde stehen in der Makro-Perspektive im Fokus der Betrachtung. Die Struktur von Organisationen ist Gegenstand für Makro-Ansätze. Ansätze für die Mikro-Perspektive sollten sich an Handlungen von Individuen orientieren, sowie den Interaktionen zwischen den Individuen. Auf der Mikro-Perspektive wird untersucht, wie Menschen typischerweise unter bestimmten Bedingungen handeln und welche Motive und Erwartungen sie dabei verfolgen. Im Mittelpunkt der Betrachtung stehen Ansätze, die vermittelnde Positionen zwischen Mikro- und Makro-Perspektive einnehmen. Sie stellen das Verhältnis zwischen Individuum und kollektiven Gebilden bzw. Strukturen dar [Rolf 2003b].

Daraus schließe ich, dass die Meso-Perspektive mit Ihrer akteursbezogenen Sichtweise für die beschriebenen softwaretechnischen Ansätze eine Schnittsstelle bietet. Diese Perspektive legt den Fokus auf die Herstellung und Anwendung von Informatiksysteme in Organisationen (siehe Kapitel 3.2). Auf dieser Ebene können Probleme durch Interessenkonflikte und Konkurrenzen der Akteure entstehen. Einige der in Kapitel 4 beschriebenen softwaretechnische Ansätze bieten unterschiedliche Vorgehensweisen an, frühzeitig Missverständnisse zu vermeiden. Beispiele hierfür sind Glossare, die Fachbegriffe des Anwendungsbereichs enthalten oder auch Prototypen, die Systemvisionen darstellen und so rechtzeitig Fehler aufzeigen. Use Cases sind

beispielsweise ein Darstellungsmittel, mit dem die Tätigkeiten der Akteure dargestellt und analysiert werden.

Aus den oben genannten Gründen ist die Meso-Perspektive von entscheidender Bedeutung für die softwaretechnischen Ansätze.

Im Folgenden werden die einzelnen softwaretechnischen Ansätze in dem MICROPOLIS-Modell betrachtet.

Softwaretechnischer Ansatz	MICROPOLIS-Modell		
	Mikro-Perspektive	Meso-Perspektive	Makro-Perspektive
	Einzelne Akteure	Akteure & Arenen	Globale Gesellschaft
Objektorientierte Analyse	+	++	
Anforderungsermittlung im Requirements Engineering	+	++	
Contextual Design	+	++	
Extreme Programming		+	
STEPS (Softwaretechnik für Evolutionäre Partizipative Software)	+	++	+

Tabelle 5-1: Gegenüberstellung der softwaretechnischen Ansätze zu dem MICROPOLIS-Modell (Legende: + auf der Ebene anzuordnen, ++ Hauptebene, auf der sich der Ansatz anordnen lässt.)

Tabelle 5-1 zeigt die Gegenüberstellung der softwaretechnischen Ansätze zu dem MICROPOLIS-Modell und bietet eine Übersicht, welcher softwaretechnische Ansatz auf welcher Ebene und mit welcher Perspektive beschrieben werden kann.

Objektorientierte Analyse

Die Objektorientierte Analyse befasst sich mit den Gegenständen der Anwenderwelt und modelliert diese für den Entwurf bei der Softwareentwicklung. Die Analyse betrachtet sowohl den einzelnen Akteur und seinen Arbeitsplatz, als auch Akteursgruppen. Aus diesen Gründen kann die Objektorientierte Analyse mit der Mikro und der Meso-Perspektive beschrieben werden.

Die Unified Modelling Language (UML), die in die objektorientierte Analyse integrierte graphische Darstellungsform, bietet dem MICROPOLIS-Modell eine Möglichkeit, die Akteursbeziehungen zu gestalten.

Anforderungsermittlung im Requirements Engineering

Das Requirements Engineering beinhaltet die Anforderungsanalyse genauso, wie die Maßnahmen, die die Analyse und die weitere Verwendung unterstützen [Rupp 2002]
Das Requirements Engineering hat somit viele Aufgaben und bildet einen eigenständigen Bereich in der Softwareentwicklung. Die Anforderungsermittlung; eine der Aufgaben im Requirements Engineering, ist die Kunst, Anforderungen so zu sammeln, dass sie für den Entwurf und die Planung der Softwareentwicklung genutzt werden können [Rupp 2002].
Die Anforderungsermittlung ist auf der Meso-Perspektive anzusiedeln.

Contextual Design

Contextual Design ist eine Gestaltungsmethodik mit dem Fokus auf den Kontext der Benutzer und deren Arbeit. Es dient dem MICROPOLIS-Modell auf der Ebene der einzelnen Akteure, sowie auf der Ebene der Akteursgruppen.

Benutzerrelevante Themen und Informationen werden im Contextual Design in einem wandgroßen, hierarchischen Diagramm dargestellt. Um übergreifende Strategien und Ziele aufzudecken, werden die einzelnen Arbeitsmodelle in konsolidierten Modellen zusammengeführt. Dadurch werden gleichzeitig individuelle Unterschiede organisatorisch sichtbar und berücksichtigt [Holzblatt 1999].

Skizzen auf Papier dienen dieser Gestaltungsmethode als Werkzeug (siehe Kapitel 4.3 Contextual Design). Hier ist der Anknüpfungspunkt des Contextual Design an das MICROPOLIS-Modell. Das MICROPOLIS-Modell kann mit der Mikro-, als auch mit der Meso-Perspektive das Contextual Design betrachten. Gleichzeitig besteht die Möglichkeit für das MICROPOLIS-Modell, sich der Gestaltungsart des Contextual Design bedienen. Die graphische Darstellung erfolgt mittels Kreisen, Rechtecken, Pfeilen und textuellen Beschreibungen. So könnten auch im MICROPOLIS-Modell die Akteursbeziehungen modelliert, verdeutlicht und gestaltet werden. Das Contextual

Design betrachtet den Kontext der Arbeit und nicht den Kontext zu anderen Firmen oder der globalen Gesellschaft. Im MICROPOLIS-Modell kann dennoch die Gestaltungsart auf die oberste Perspektive übertragen werden. Die Globale Gesellschaft (siehe Kapitel 3.3 Makro-Perspektive) besteht aus Institutionen, Werten und Leitbildern. Man kann die einzelnen Insitutionen der globalen Gesellschaft so betrachten, als seinen diese die einzelnen Akteursgruppen einer Organisation. Die Organisation, namens „globale Gesellschaft“ hat verschiedene Einflussfaktoren in Form von Werten und Leitbildern. Diese könnten in einem Einfluss-Modell nach dem Contextual Design dargestellt werden.

Extreme Programming

Extreme Programming ist eine agile Entwicklungsmethodik mit zwölf Techniken (siehe Kapitel 4.4 eXtreme Programming). Die Techniken orientieren sich stark an der Zusammenarbeit von Entwickler und Kunden. In XP-Projekten werden verschiedene Rollen eingenommen, in denen die beteiligten Akteure ihre festen Aufgaben haben.

STEPS

Die Softwaretechnik für Evolutionäre Partizipative Software greift auf allen Perspektiven des MICROPOLIS-Modells, da es den Lernprozess im Fokus der Arbeit hat. STEPS geht von einem Konsens der Akteure aus.
Aus diesem Grund ist eine Integration von STEPS in das MICROPOLIS-Modell sinnvoll.
Zum besseren Verständnis, soll STEPS in den einzelnen Perspektiven demonstriert werden.
Beginnend mit der Mikro-Perspektive kann ein Zusammenhang zwischen dem MICROPOLIS-Modell und STEPS hergestellt werden, da an dieser Stelle die Aufgaben der einzelnen Akteure fokussiert werden.
Wie die anderen Ansätze, lässt sich STEPS auch auf der Meso-Perspektive anwenden, da es sich um einen Ansatz zur Softwareentwicklung handelt.
Der Gegensatz zu den anderen Ansätzen ist, dass STEPS auch auf der Makro-Perspektive zu verorten ist. Die Softwareentwicklung ist bei STEPS ein Lernprozess. Auf der Makro-Perspektive wird die globale Gesellschaft (siehe Abbildung 3-2) mit den Regeln der Institutionen auf der einen Seite und den Werten und Leitbildern auf der

anderen Seite betrachtet. Wird das MICROPOLIS-Modell nun aus einer Meta-Ebene betrachtet, in dem die Universität als eine Institution und das MICROPOLIS-Modell als Leitbild des Wirtschaftsinformatikers angesehen wird, so kann STEPS den Lernprozess unterstützen.

5.2 Wie könnte eine Integration von STEPS und MICROPOLIS-Modell gelingen?

Die Grundlage in der Informatik ist die Formalisierung, die auf der ersten Ebene, der Mikro-Perspektive anzuordnen ist.

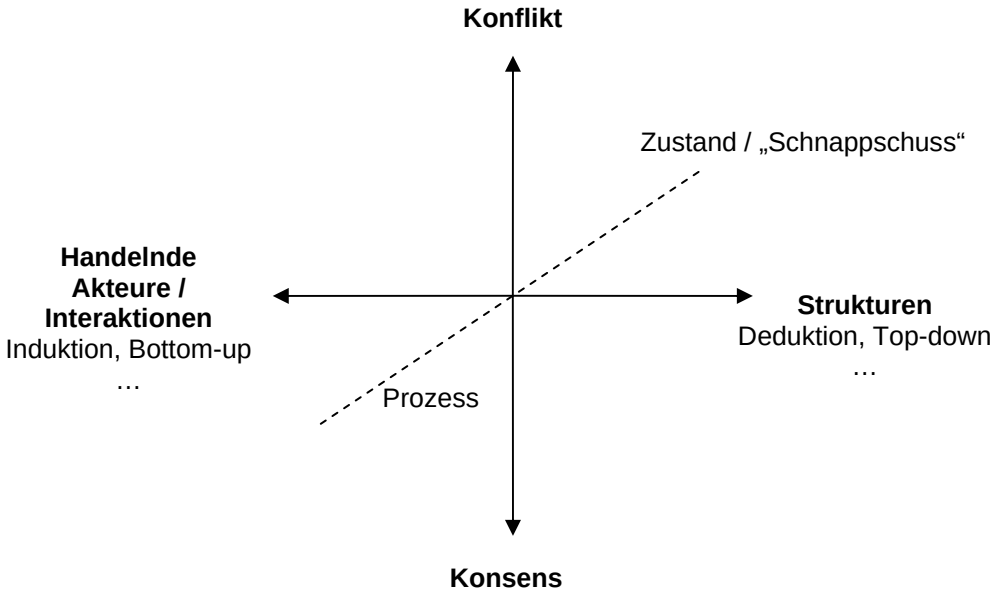

Abbildung 5-1: Klassifikation von Burell/Morgan aus den Sozialwissenschaften [Rolf 2002], S.43

In Abbildung 5-1 werden anwendungsnahe Teilgebiete der Informatik in den Klassifizierungsvorschlag von Burell/Morgan aus den Sozialwissenschaften eingeordnet. Auf einer dreidimensionalen Matrix werden die Schwerpunkte der Teildisziplinen der angewandten und praktischen Informatik zugeordnet. Die Schwerpunkte liegen innerhalb dreier Achsen. Auf der horizontalen Achse findet sich der Bereich der handelnden Akteure bzw. Interaktionen oder der Bereich der Strukturen. Auf der vertikalen Achse befinden sich die beiden Pole Ordnung bzw.

Konsens und Konflikt. Auf einer diagonalen Achse können Zustands- oder Prozess-Betrachtungen eingeordnet werden [Rolf 2002].

Für eine Einordnung wird STEPS im Folgenden innerhalb der dreidimensionalen Matrix aus der Abbildung 5-1 zugeordnet werden.

1. *Handelnde Akteure bzw. Interaktionen oder Strukturen in einer Organisation?*

   Wie in Kapitel 4.5 beschrieben ist der Blick der evolutionären partizipaktiven Softwareentwicklung STEPS auf die einzelnen Akteure mit deren Interaktionen gerichtet.

   Dies geht aus dem Namen des softwaretechnischen Ansatzes hervor. In dem Kürzel STEPS sind die Worte „partizipative“ und „Softwareentwicklung“ enthalten. Dies bedeutete eine Zusammenarbeit der beteiligten Personen an der Softwareentwicklung

2. *Ordnung bzw. Konsens oder Konflikt ?*

   Mögliche Konflikte zwischen den beteiligten Akteuren werden nicht betrachtet.
   Die Schlussfolgerung hieraus ist, das der Ansatz STEPS von dem Konsens ausgeht und in dem Koordinatensystem auf dem entsprechenden Pol anzusiedeln ist.
   Ganz im Gegensatz zum MIKROPOLIS-Modell, in dem Konflikte und auch die Ursachen für Konflikte in den unterschiedlichen Interessen oder Kommunikationslücken gesehen werden.

3. *Zustands – oder Prozess – Sicht?*

   In STEPS gibt es die produktorientierte und die prozessorientierte Sichtweisen (siehe Kapitel 4.5).

   Bei der produktorientierten Sichtweise wird Software als ein eigenständiges Produkt betrachtet, welches aus Programmen und definierten Dokumenten besteht. Hier ist eine klare Trennung der Softwareentwicklung vom Einsatz der Software zu sehen. Die Softwareentwicklung wird hier als Herstellung des Produktes „Software“ angesehen.

Im Gegensatz dazu sieht die prozessorientierte Sichtweise die Software in einem Kontext von Prozessen (Lern-, Arbeits- oder Kommunikationsprozesse), aus dem das Produkt Software hervorgeht. Bei dieser Sichtweise werden die Herstellung und der Einsatz von Software gemeinsam und miteinander verschränkt betrachtet.

Die Verbindung der produktorientierten und der prozessorientierten Sichtweisen liegt darin, das die Prozesse zur Softwareentwicklung (prozessorientiert) das Produkt Software (produktorientiert) hervorbringen.

Hieraus folgt, dass der Schwerpunkt von STEPS auf der „Prozess – Sicht“ Achse liegt.

Betrachtet man nun diese drei Dimensionen, so ist STEPS in dem Koordinatensystem zwischen den Schwerpunkten „Interaktion“, „Konsens“ und „Prozess“ (siehe Abbildung 5-2, optisch unten links) einzuordnen. Die konstruktive Softwaretechnik und die Wirtschaftsinformatik sind dem gegenüber einzuordnen, d.h. zwischen den Schwerpunkten „Struktur“, „Konflikt“ und „Zustand“.

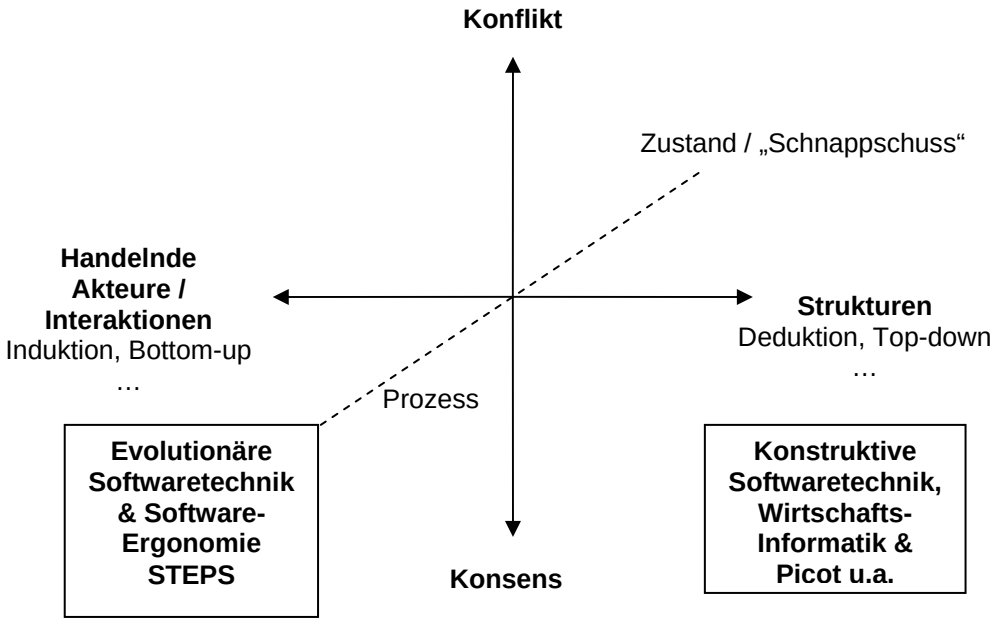

Abbildung 5-2: Klassifikation Einordnung anwendungsnaher Teilgebiete der Informatik (siehe [Rolf 2002], S. 47)

STEPS geht von Akteuren Bottom-up (siehe Abbildung 5-12) aus und betrachtet nicht die Gesamtorganisation Top-down. Im MICROPOLIS-Modell sagt die Meso-Perspektive mit Akteuren und Arenen aus, dass es zu Konflikten, aufgrund unterschiedlicher Interessen kommen kann. Hier ist der Hauptunterschied zu dem Vorgehensmodell STEPS, das von einem gemeinsamen Konsens ausgeht und Konflikte nicht beachtet.
STEPS eignet sich gut für die Meso-Perspektive, weil es aufzeigt, wie Software mit Unterstützung der Akteure eingeführt werden kann.
STEPS vernachlässigt aber das Management und übersieht, dass es feste Handlungen in Organisationen gibt.

Um das Vorgehensmodell STEPS in das MICROPOLIS-Modell zu integrieren, werden Methoden zur Konfliktlösung benötigt.
Zwischen dem MICROPOLIS-Modell und dem Methodenrahmen STEPS gibt es folgende Gemeinsamkeiten. Zum einen liegen beiden gemeinsame Begriffe zu Grunde, zum anderen die Sicht auf die im Prozess beteiligten Personen, den Akteuren.
Beiden Ansichten betrachten den Prozess in der Softwareentwicklung. Der Einsatz der IT beeinflusst die Arbeit verschiedener beteiligter und betroffener Personengruppen. Die Akteure sind in Strategien und Aktionen anderer Akteure eingebunden, so dass die Interessen vieler ebenfalls beeinflusster Gruppen berücksichtigt werden müssen. Ziel bei der Softwareentwicklung ist es, eine Lösung zu finden, die für die beteiligten Akteure passend ist und die erforderlichen Aufgaben unterstützt.

Über Leitbilder und Metaphern werden Wertvorstellungen der Akteure ausgedrückt und können so in den Entwicklungsprozess übernommen werden [Rolf 1998].

6 Schlussbetrachtung

In diesem Kapitel werden die Ergebnisse der Diplomarbeit zusammengefasst und auf die markanten und deutlich gewordenen kritischen Punkte eingegangen. Das Kapitel schließt mit einem Ausblick für die Zukunft ab.

6.1 Zusammenfassung

Der primäre Fokus des MICROPOLIS-Modells liegt auf dem Aspekt, die Informatik in organisatorische Kontexte der Gesellschaft einzuordnen. Aus diesem Grund dient es der Ausbildung von Informatikern und Wirtschaftsinformatikern, die zusätzlich zu dem Verfügungswissen auch Orientierungswissen vermittelt bekommen. Es handelt sich um einen Orientierungsrahmen, der der Wirtschaftsinformatik als Leitbild dienen kann.

Im MICROPOLIS-Modell wird die Informatik aus den Mikro-, Meso und Makro-Perspektiven fokussiert. In der Mikro-Perspektive wird der Kern der Informatik betrachtet. Die Meso-Perspektive bezieht die Umgebung der Informatik, d.h. Organisationen und Akteure mit ein. Die Makro-Perspektive zieht zusätzlich die Globale Gesellschaft in den Blickwinkel der Betrachtung.

Die Aufgabe und das Ziel dieser Diplomarbeit ist es, ausgewählte softwaretechnische Methoden zu untersuchen und so mit dem MICROPOLIS-Modell zu verknüpfen, dass mögliche Orientierungs- und Gestaltungsperspektiven aufgezeigt werden.

Zusammenfassend betrachtet, bietet das MICROPOLIS-Modell mit seiner Sichtweise auf Organisationssysteme in Organisationen

- eine Orientierungssicht
- steht in einem gesellschaftlichen und organisatorischen Kontext und
- steht der Wirtschaftsinformatik als Leitbild zu Verfügung

Die Softwaretechnik (SWT) auf der anderen Seite

- liefert die Ansätze in Form von Methoden, Techniken und Modelle und
- bietet eine Gestaltungsperspektive

Das MICROPOLIS-Modell bietet der Informatik, der Wirtschaft, als auch der Gesellschaft eine Orientierungssicht. Diese drei verschiedenen Bereiche werden in einen gesellschaftlichen und organisatorischen Kontext gesetzt. Unter Einbeziehung verschiedener Perspektiven (Mikro, Meso und Makro-Perspektive), werden die unterschiedlichen Disziplinen in einen Zusammenhang gesetzt und verbunden.

Die Softwaretechnik, die sich mit der Erstellung von Softwareprodukten beschäftigt, ist auf der mittleren Ebene, der Meso-Perspektive anzusiedeln. Mit ihren Methoden und Techniken bieten sich dem MICROPOLIS-Modell eine Gestaltungsmöglichkeit und Anknüpfungspunkte für weitere Entwicklungen.

Der softwaretechnische Ansatz STEPS ist auf allen Perspektiven des MICROPOLIS-Modells zu verorten und bietet somit auch Anknüpfungspunkte. Zudem zeigt STEPS wie Software mit Unterstützung der Akteure eingeführt werden kann.

Das MICROPOLIS-Modell kann sich zur Darstellung der einzelnen Perspektiven verschiedener Techniken der in dieser Arbeit genannten Methoden bedienen.

6.2 Kritische Würdigung

Die Kritische Würdigung beinhaltet sowohl die problematischen Punkte zwischen dem MICROPOLIS-Modell und die für diese Arbeit ausgewählten softwaretechnischen Ansätze, als auch die Möglichkeiten, die sich durch eine solche Verbindung bieten.
Strukturell kann es zwischen dem MICROPOLIS-Modell und den softwaretechnischen Ansätzen Probleme geben. Diese Probleme entstehen durch die Forderungen an die softwaretechnischen Ansätze. Die für diese Arbeit ausgewählten softwaretechnischen Ansätze erfüllen in einem Softwareentwicklungsprojekt die Aufgabe, die verschiedenen Entwicklungsphasen zu unterstützen. Dies geschieht anhand von Richtlinien und Vorschriften, Mitteln und Werkzeugen.

In verschiedenen softwaretechnischen Ansätzen werden Leitbilder und Metaphern verwendet (siehe Kapitel 4). Da das MICROPOLIS-Modell der Wirtschaftinformatik als ein Leitbild bietet, kann dieses in der Softwareentwicklung eingesetzt werden. Mit dem Orientierungswissen, das durch das MICROPOLIS-Modell vermittelt wird kann

6.3 Ausblick

Die softwaretechnischen Ansätze können für das MICROPOLIS also nicht einfach übernommen werden. Das MICROPOLIS-Modell muss auf einer Meta-Ebene die globale Gesellschaft betrachten und kann sich dann der einzelnen Techniken bedienen.

Das MICROPOLIS-Modell betrachtet die Konflikte, die zwischen Akteuren entstehen können. Dies bietet den softwaretechnischen Ansätzen die Möglichkeit, Auswirkungen von Handlungen zu betrachten und die realen Bedingungen zu analysieren.

Auf der Makro-Perspektive wird die globale Gesellschaft (siehe Abbildung 3-2) mit den Regeln der Institutionen auf der einen Seite und den Werten und Leitbildern auf der anderen Seite betrachtet.

Abschließen ist zu beachten, dass ausgehend von einer Meta-Ebene, die Universität und das MICROPOLIS-Modell selbst auf der Meso-Perspektive betrachtet werden können. Dies bedeutet, dass der Fachbereich Universität als eine Institution gesehen würde. Die Institutionen sind geprägt durch die Leitbilder und Regeln. Das MICROPOLIS-Modell kann als Leitbild des Wirtschaftsinformatikers angesehen werden. In diesem Fall unterstützt STEPS den Lernprozess der Institution Universität, wie bereits in den Kapiteln 4.5 und 5 erläutert.

7 Abkürzungsverzeichnis

AM	Anforderungsmanagement
ASI	Angewandte und sozialorientierte Informatik
BDV	Betriebswirtschaftliche Datenverarbeitung
BWL	Betriebswirtschaftslehre
CD	Contextual Design
GI	Gesellschaft für Informatik
ISO	Informatiksysteme für Organisationen
Makro-Perspektive	Oberste Schicht im MICROPOLIS-Modell
Meso-Perspektive	Mittlere Schicht im MICROPOLIS-Modell
MICROPOLIS-Modell	„Microelectronic“ und „polis“ [altgriech. Stadtstaat]
Mikro-Perspektive	Unterste Schicht im MICROPOLIS-Modell
OOA	Objektorientierte Analyse
OWI	Orientierungs- und Wirtschaftsinformatik
RE	Requirements Engineering
STEPS	Softwaretechnik für Evolutionäre Partizipative Softwareentwicklung
SWT	Softwaretechnik
UML	Unified Modelling Language
XP	eXtreme Programming

8 Literatur

[Andersen et al. 1990] Andersen, N. E.; Kensing, F., Lundin, J., Mathiassen, L; Munk-Madsen, A.: Rasbech, M.; Sogaard, P.: „Professional system development“, Prentice Hall, New York 1990

[Balzert 1999a] Balzert, H.: „Lehrbuch der Objektmodellierung. Analyse und Entwurf.“, Spektrum Akademischer Verlag, Heidelberg 1999.

[Balzert 1999b] Balzert, H: „UML kompakt“, Spektrum Akademischer Verlag, Heidelberg 1999.

[Beck 2000] Beck, Kent: „Extreme Programming Explained – Embrace Change, Reading, Addisson-Wesslay, Massachusetts 2000

[Coy et al. 1992] Coy, Wolfgang; Nake, Frieder; Pflüger, Jörg-Martin; Rolf, Arbo; Seetzen, Jürgen; Siefkes, Dirk; Stransfeld, Reinhard(Hrsg.):“Sichtweisen der Informatik“; Friedr. Vieweg & Sohn Verlagsgesellschaft, Braunschweig 1992

[Beyer und Holzblatt 1998] Beyer, Hugh; Holtzblatt, Karen: „Contextual Design - Defining Customer-Centered-Systems.” Morgan Kaufman, 1.Auflage, San Francisco 1998

[Duden 1990] Drosdowski, Günther (Hrsg.): „Das Fremdwörterbuch - Duden Band 5“, erw. Auflage, Duden Verlag, Mannheim 1990

[Friedrich 1995] Friedrich, Jürgen; Herrmann, Thomas; Peschek, Max; Rolf, Arno (Hrsg.):“Informatik und Gesellschaft“, Spektrum Akademischer Verlag, Heidelberg 1995

[Floyd et al. 1990] Floyd, Christiane; Mehl, M.; Reisin, F.-M.; Schmidt, G.; Wolf, G.: „Projekt PEts: Partizipative Entwicklung transparenzschaffender Software für EDV-gestützte Arbeitsplätze”, Technische Universität Berlin, Forschungsgruppe SWT, Berlin 1990

[Floyd 1992] Floyd, Cristiane: „STEPS-Projekthandbuch“, Universität Hamburg, 1992

[Floyd 1994] Floyd, Christiane: „Software-Engineering – und dann?“, In: „Informatik Spektrum, S. 29-37, Band 17, Springer-Verlag, Berlin 1994“

[Floyd 1995] Floyd, Christiane: „Software Engineering: Kritik und Perspektiven“ In: [Friedrich 1995], S.238-245

[Floyd und Züllighoven 1997] Floyd, Christiane; Züllighoven, Heinz: Was ist Softwaretechnik? In: Rechenberg und Pomberger: Informatik Handbuch. Hanser Verlag, München 1997, Seiten 641-667 In: [Rechenberg und Pomberger 2002], S.763-790

[Floyd und Oberquelle 2003a] Floyd, Christiane; Oberquelle, Horst: „Softwaretechnik und Software-Egonomie- Skript zur Vorlesung Wintersemester 2003/2004“, Teil 1, Universität Oktober 2003

[Floyd und Oberquelle 2003b] Floyd, Christiane; Oberquelle, Horst: „Softwaretechnik und Software-Egonomie- Skript zur Vorlesung Wintersemester 2003/2004“, Teil 2, Universität Oktober 2003

[Floyd und Pape 2004] Floyd, Christinane; Pape Bernd: „Softwareentwicklung als Wissensprojekt – am Beispiel der Commsy-Entwicklung“, Universität Hamburg 2004 In: „Pape, Bernd., Krause, Detlev., Oberquelle, Horst. (Hrsg.): Wissensprojekte - gemeinschaftliches Lernen aus didaktischer, softwaretechnischer und organisatorischer Sicht, Waxmann Verlag GmbH, Münster 2004, S.303 - 329“

[Hesse et al. 1994] Hesse W.; Barkow, G.; von Braun, H.; Kittlaus, H.-B. und Scheschonk: „Terminologie der Softwaretechnik“ In: „Informatik Spektrum, S. 39 - 47, Band 17, Springer-Verlag, Berlin 1994“

[Holzblatt 1999] Holzblatt, Karen: „Customer Centred Design as Discipline“, 1999 In: Sasse, M.A., Johnson, C(Eds): Human-Computer interaction – INTERACT ’99, IOS Press, 1999, S. 3-17

[Informatik-Begriffsnetz 2005] Das Informatik-Begriffsnetz von der Gesellschaft für Informatik e.V. (GI) : Fachgruppe "Test, Analyse und Verifikation von Software" (TAV); Arbeitskreis Software-Qualität in Franken (ASQF) http://www.vorgehensmodelle.de/Giak/index.html Stand: Januar 2005

[Lippert et al. 2002] Lippert, Martin; Roock, Stefan; Wolf, Henning: „Software entwicklen mit eXtreme Programming – Erfahrungen aus der Praxis“, dpunkt.verlag, Heidelberg 2002

[Mittelstrass 1989] Mittelstrass, Jürgen: „Glanz und Elend der Geisteswissenschaften“, Oldenburg : Bibliotheks- und Informationssystem der Univ., Oldenburg 1989 http://docserver.bis.uni-oldenburg.de/publikationen/bisverlag/unireden/ur27/dokument.pdf

[Rolf und Siefkes 1992] Rolf, Arno; Siefkes, Dirk: “Wozu Grundlagen ? (Einleitung)“, 1992 in [Coy et al. 1992], S. 13-16

[Rolf 1998] Rolf, Arno: Grundlagen der Organisations- und Wirtschaftsinformatik, Springer-Verlag, Berlin, Heidelberg 1998

[Rolf 1999] Rolf, Arno: „Von Leitbildern, Moden und Langfristzielen der Wirtschaftsinformatik“, Wirtschaftsinformatik 41 (1999) 2, S. 184-186

[Rolf 2002] Rolf, Arno: „Informatiksysteme in Organisationen“, Miteilung 317, Universität Hamburg Juni 2002

[Rolf 2003] Rolf, Arno.: „Skript zur IMG-Vorlesung im Sommersemester 2004“, Universität Hamburg 2003

[Rolf 2003b] Rolf, Arno: „Ausgangspunkte für eine Gestaltungslehre Informatiksysteme in Organisationen und Gesellschaft“, Universität Hamburg, vervielfältigtes Manuskript 2003

[Rolf 2004a] Rolf, Arno: „Mitteilung 330 – Informatiksysteme in Organisationen und Gesellschaft“ Teil A: Informatik in Organisationen und globaler Ökonomie –Ein Orientierungsrahmen“, Universität Hamburg: März 2004

[Rolf 2004b] Rolf, Arno: „Informatiksysteme in Organisationen und Globalen Gesellschaften IMG 4-Skript“, WS 2004/2005, Universität Hamburg 2004

[Rupp 2002] Rupp, Chris: „Requirements– Engineering und –Management – professionelle, iterative Anforderungsanalyse für IT-Systeme", HANSER, München 2002

[Schefe 1999] Schefe, Peter: „Softwaretechnik und Erkenntnistheorie" In: „Informatik Spektrum, S. 122-135, Band 17, Springer-Verlag, Berlin 1994"

[Schienmann 2002] Schienmann, Bruno: „Kontinuierliches Anforderungsmanagement" Addison-Wesley, München 2002

[Volpert 1992] Volpert, Walter:" Erhalten und Gestalten. Von der notwendigen Zähmung des Gestaltungsdrangs, in [Coy et al. 1992]", S.171-180

[Züllighoven 1998] Züllighoven, Heinz: „Das objektorientierte Konstruktionshandbuch nach dem Werkzeug & Materialansatz", korrigierter Nachdruck, Heidelberg: dpunkt.verlag 1998

Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbständig und ohne fremde Hilfe angefertigt habe und mich anderer als im beigefügten Verzeichnis angegebener Hilfsmittel nicht bedient hab. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

Datum, Natalie Daou